Huslik

# CPC464 inside out

Ein Buch für Programmierer

# CPC464
inside out

Huslik

# CPC464
# inside out

Ein Buch für Programmierer

ROM-Listing mit stack-orientierter Darstellung,
mit Referenz-Adressen, ausführlich kommentiert.

Hinweis:

Die neben dem Source-Code enthaltenen Kommentare stellen, sofern es sich nicht um veröffentlichte Einsprungadressen handelt, die persönliche Meinung des Buchautors dar, die nicht notwendigerweise mit der des Progammautors übereinstimmen muß. Eine Gewähr für die Richtigkeit der angegebenen Funktionen, Adressen und sonstigen Angaben wird nicht übernommen. Der Verlag übernimmt keine Verantwortung für die Nutzung dieser Informationen, insbesondere nicht für die Verletzung von Patent- und anderen Rechten Dritter, die daraus resultieren. Der Hersteller kann Programme, Programmteile oder Einsprungadressen ohne vorherige Ankündigung ändern. Für verbindliche Informationen wird der Leser an den Hersteller verwiesen.

Quellen:

AMSOFT, CPC464 Operating System Firmware Specification
AMSOFT/SCHNEIDER, CPC464 Benutzer-Handbuch
Amstrad Consumer Electronics plc., CPC464 ROM Data
ZILOG, Z80 CPU Technical Manual

Z80 ist ein eingetragenes Warenzeichen der Firma ZILOG

ISBN 3-925159-00-2



Printed in Germany
Fotosatz: Stempelfabrik Huslik, Augsburg
Druck: MARODRUCK, Augsburg

# Vorwort

Was dürfen Sie von diesem Buch erwarten? Mit Hilfe der Kommentare sollte es Ihnen möglich sein, die wichtigsten Abläufe und Funktionen im Betriebssystem und im Basic zu verstehen. Eine Vielzahl von Informationen, zum Beispiel die Syntax bei Basic-Anweisungen oder die Angabe der verwendeten Register, soll Sie dabei unterstützen. Die stack-orientierte Darstellung (das Ein- und Ausrücken bei Push- und Pop-Befehlen) macht die Programme lesbarer und betont die Stellen des Programms, an denen echte Aktionen stattfinden. Die bei jeder Routine angegebenen Referenzadressen ermöglichen es, das Programm auch nach rückwärts zu verfolgen.

Dieses Buch kann und soll weder ein Handbuch noch eine Hardware-Beschreibung ersetzen. Bei der Komplexität der Zusammenhänge ist es ohnehin ratsam, so viele Informationsquellen wie nur möglich zu Rate zu ziehen.

Ich hoffe, daß es mir gelingt, Ihnen mehr als ein freudiges 'Aha' zu entlocken!

# Inhalt

## Wie es dazu kam

Als ich im November 1984 den in zwei Kartons verpackten Kleincomputer aus dem Kaufhaus nach Hause trug, wußte ich noch nicht so genau, was ich da gekauft hatte. Ich dachte eigentlich zunächst nur an das, was man sehen konnte: Ein Computer mit Tastatur und Bildschirm, grün, bei dem man nach den Prospektangaben die Tastaturbelegung und sogar einzelne Zeichen selbst definieren kann. Angenehm und kaufentscheidend waren die Zehnertastatur und die Cursor-Tasten (und natürlich der Preis). Für den vorgesehenen Zweck, als intelligentes Erfassungsterminal, glaubte ich es allemal gut einsetzen zu können. Zur Datenspeicherung war ich zwar Besseres gewohnt, doch dachte ich, zunächst tut's der ohnehin bereits eingebaute Cassettenrecorder auch - und - die Floppy soll ja noch kommen. Zu Hause machte ich mich erst einmal mit dem recht umfangreichen Handbuch vertraut, probierte ein paar Beispiele und mein Erstaunen und mein Respekt vor diesem kleinen Gerät wuchs von Stunde zu Stunde. Dabei war es nicht einmal der Sound oder die Grafik; was mich faszinierte, war das ganz 'normale' Basic, die langen Variablennamen, die Möglichkeit, in Basic 'Multitasking' zu veranstalten, Tasten zu belegen, Zeichen zu definieren und eine Reihe von weiteren Details, auf die ich gerne noch eingehen werde.

Schließlich wollte ich es dann ganz genau wissen. Ich versuchte am nächsten Tag - leider vergeblich - die im Handbuch erwähnte Hardware-Beschreibung zu erhalten. So begnügte ich mich zunächst mit 'Tips & Tricks' aus dem Hause Data-Becker-Verlag.

An dieser Stelle sei - in aller Bescheidenheit - erwähnt, daß ich bereits fast zehn Jahre Erfahrung mit Basic, Fortran und Assembler auf einem Mini (einer NOVA2 von Data General) habe und vor ein paar Jahren auch Basic und Assembler auf einem CBM 8032 programmiert habe. Vom Z80 (oder 8080) hatte ich jedoch noch keine Ahnung. Doch dank Rodnay Zaks' hervorragendem Handbuch aus dem Sybex Verlag konnte auch diese Wissenslücke bald geschlossen werden.

Doch nun ging's ans Innenleben. Als Programmierer möchte man natürlich wissen, wo liegt das Basic, wo werden die Variablen gespeichert, wo liegen die Informationen, die beim 6502 in der Page Zero liegen, wo sind die Zeichen definiert, wo die Daten über die Tastaturbelegung, welche Bereiche und Adressen werden vom Betriebssystem und vom Basic belegt oder welche stehen für eigene Zwecke zur Verfügung. Von meiner NOVA und vom CBM war ich es gewohnt, vorhandene Routinen zu nutzen, so lag der Wunsch nahe, dies auch beim CPC464 zu tun.

Zunächst wurde mal ein einfacher Hexdump erzeugt, um vielleicht schon an den enthaltenen lesbaren Texten 'etwas zu erkennen'. Die unbefriedigende Erkenntnis, wo die wenigen lesbaren Texte lagen, führte sehr schnell dazu, aus dem Hexmonitor einen richtigen Disassembler zu entwikkeln. Dies war die Gelegenheit, den, aus der Bit-Ebene gesehen, recht holprigen Befehlssatz des Z80 genau kennenzulernen.

So wichen die vielleicht 50 Seiten Hexdump zunächst ca. 250 Seiten Z80-Code, natürlich ohne Kommentare und mit all den Ungereimtheiten die daraus resultieren, daß man ja wirklich noch keine Ahnung hat, wo Daten stehen, oder wie manche Unterroutinen ihre Parameter erhalten. Freilich mußte auch der Versuch, sozusagen von Adresse Null ausgehend, nachzuvollziehen, 'was er denn so macht' und dabei Kommentare in das Listing zu schreiben, als völlig aussichtslos aufgegeben werden, vor allem deshalb, weil man bei den immer weiter verschachtelten Unterprogrammaufrufen bald nicht mehr weiß, wo vorne und hinten ist.

So ging ich dann die Sache von drei Seiten her gleichzeitig an. Einmal versuchte ich, den Sinn kleiner einfach gestalteter Unterroutinen zu entschlüsseln (das nennt man 'bottom up'), zum anderen hoffte ich vom Basic her, dessen Kommandos und Funktionen noch relativ leicht zu lokalisieren waren, die Verwendung der Sprunglisten und Betriebssystem-Routinen zu erkennen. Natürlich hatte ich auch eine - zugegebenermaßen zunächst noch etwas nebelhafte - Vorstellung von dem, was in so einem Rechner notwendigerweise ablaufen muß. Und irgendwie und irgendwo mußte das auch ablaufen (bekannt unter dem Begriff 'top down').

Schließlich konnte ich mir dann von England die hier noch nicht erhältliche Hardware-Beschreibung beschaffen (inzwischen ist auch die deutsche Übersetzung der Firmware-Beschreibung erschienen und wird dringend als Arbeitsunterlage empfohlen), die dem immer noch mageren Gerippe schon etwas Fleisch verschaffte.

Nicht, daß Sie jetzt aber glauben, mit dem Handbuch wäre das Listing bereits komplett! Die ca. 200 Einsprung-Adressen gaben dem ganzen zwar einen gewissen Halt, doch waren diese im Vergleich zu den jetzt weit über 6000 Kommentarzeilen und Einzelanweisungen nur ein bescheidener Anfang, und was das Basic betrifft, so ist darüber nichts in der Firmware-Beschreibung enthalten.

Wenn also die normalen Methoden nicht halfen, mußte eine bessere gefunden werden. Der Disassembler wurde nun so aufgebaut, daß er während des Ablaufs die 'Erkenntnisse' über eine bestimmte Speicherstelle oder Routine automatisch dann einbaut, wenn auf diese Bezug genommen

wird. Die 'Erkenntnisse' standen nun zunächst in Data-Anweisungen. Immer neue Erkenntnisse führten aber bald dazu, daß der Platz im Speicher nicht mehr ausreichte. Und so war ich gezwungen, den Disassembler mit allen Erkenntnissen samt ROM-Inhalt auf meine NOVA mit einem Magnetplattensystem zu übertragen. Die Übertragung klappte auch ganz gut, nachdem ich eine ohnehin nicht mehr benutzte Streifenleser-Schnittstelle meiner NOVA zum parallelen Eingang umgebaut und mit dem Drucker-Ausgang des CPC464 verbunden hatte. Der in Basic vorhandene Disassembler wurde nochmal neu in FORTRAN geschrieben. Für die Ausgabe in Hex war noch eine kleine Assembler-Routine notwendig. Bis alles wieder so richtig lief, waren zwar weitere zwei Wochen vergangen, doch nun ging es zügig voran nach dem Schema: Listing ausdrucken (die 'Erkenntnisse' und '?? Vermutungen ??', die in einer eigenen Datei abgelegt waren, fügte der Disassembler bereits ein), Durcharbeiten des Listings und wieder neue Kommentare anmerken, die neuen Erkenntnisse wieder eintippen, Vermutungen berichtigen, und wieder ein neues Listing ausdrucken usw. Immer neue Anforderungen kamen auf den Disassembler zu: Ab einer bestimmten Adresse im Byte-Format ausgeben, dann wieder Z80-Code, nun folgen Adressen, Tabellen, Basic-Token, Ascii-Zeichen, Fünf-Byte-Real-Zahlen. Zuletzt mußte auch die Ausgabe noch elegant formatiert werden: Variable Seitenlängen, lebende Kolumnentitel, Seitennumerierung und Seitenumbruch.

Heute möchte ich sagen, daß etwa ein Drittel der Zeit für den Disassembler, ein Drittel für das Studium des Listings und den Rest mit reinen Schreibarbeiten drauf ging. Rund 10.000 Blatt Papier wurden bedruckt, 15 Farbbänder verschlissen und ungezählte Nächte über den Listings und am Bildschirm verbracht.

Hier muß ich anmerken, daß ich wohl kaum mehr über den CPC464 geschrieben hätte, wäre mir nicht schon bald klar gewesen, welches hervorragende Konzept hier realisiert wurde. Die Tragweite dieser Entwicklung wird einem, so glaube ich, erst bewußt, wenn man sich mit dem Detail beschäftigt. Dies und der Umstand, daß es noch keine anderweitigen Informationsmöglichkeiten gab, ließen mich den Entschluß fassen, diese gesammelten Informationen als Buch herauszugeben, um auch anderen Interessierten die Möglichkeit zu geben, den CPC besser kennenzulernen und zu nutzen. Ich hoffe, daß das Buch jetzt zum richtigen Zeitpunkt erschienen ist und daß der Stil sowie die Aufmachung des Buches angenommen wird. Es ist mir natürlich bewußt - und jeder, der sich schon einmal mit der Analyse eines Programms beschäftigt hat, wird dies gerne zugeben -, daß man ein

Programm dieses Umfangs nicht in drei Monaten bis ins letzte Detail beschreiben kann. Aus diesem Grunde habe ich die mir weniger vordringlich erscheinenden Bereiche, wie Cassettenoperationen und Sound, nur gestreift, dafür aber die Verwaltung der Variablen und Interpretationen des Basic-Programms ausführlich mit Kommentaren versehen.

In der Firmware-Beschreibung war leider nichts über die im Betriebssystem vorhandenen arithmetischen Routinen enthalten. Soweit die Funktion klar zu erkennen war, (etwa aus der Art, wie das Basic diese verwendet), wurden die Bezeichnungen angegeben, in Zweifelsfällen habe ich die für Vermutungen verwendeten '??' stehen lassen; diese Angaben sind dann ohne Gewähr. Weitere Ausarbeitungen bleiben einer künftigen Auflage vorbehalten, wobei ich für Anregungen, Kommentare und Meinungen dankbar bin. Die am Ende mitgehefteten Bögen können sie für Ihre Mitteilungen verwenden.

Vielleicht sind Sie enttäuscht, daß die Kommentare im Listing englisch abgefaßt sind, doch werden Sie beim Lesen selbst feststellen, daß Sie die Texte nicht zu übersetzen brauchen, da diese nur aus Worten bestehen, die dem normalen Jargon der Programmierer entnommen sind. Meine ersten Versuche, deutsche Kommentare anzubringen, gab ich schnell wieder auf, weil es mir einfach nicht gelingen wollte, mit wenigen Worten das Entscheidende treffend auszudrücken. Einige Begriffe, die vielleicht nicht so geläufig sind, habe ich im Anhang als kleines Wörterbuch zusammengefaßt.

Augsburg, im Februar 1985

Winfried Huslik

## Das Betriebssystem des CPC464

Wenn man einmal davon absieht, daß es nur eine einzige Interrupt-Quelle gibt, nämlich den Time-Interrupt alle 1 / 300 Sekunden, könnte man glatt von einem Mainfraim im Kompaktformat sprechen; jedenfalls dürfte das hier verwirklichte Konzept der Event-Verarbeitung einmalig sein für einen Rechner dieser Größenordnung.

Das Betriebssystem besticht durch seine klare Gliederung, sowohl was die Handhabung der Hardware betrifft, als auch die verschiedenen Steuerungsebenen. Als Beispiel mag hier die Tastatureingabe dienen. Während auf der untersten Ebene die Tastatur abgefragt, entprellt und ein Bit für eine gedrückte Taste gesetzt wird, wird auf der zweiten Ebene eine Tastennummer übergeben und auf der dritten Ebene die Tastennummer in ein Zeichen entsprechend den Tabellen für Normal-, Shift- und Control-Codes umgewandelt. Darüber wird, falls es sich bei dem von der dritten Ebene übergebenen Zeichen um ein Key-Token handelt (eine von 32 möglichen Funktionstasten), dieses zu einem String expandiert.

Auf einer noch höheren Ebene (es muß nicht notwendigerweise Nummer fünf sein) steht schließlich eine komplette Editier-Funktion zur Verfügung, die außer der Tastatur den Tastatur-Puffer, die Bearbeitung der Cursor-Steuertasten übernimmt und dabei gleichzeitig die Lage und Größe eines angesprochenen Bildschirmfensters berücksichtigt.

Über eine Sprungtabelle im RAM ab B900 stehen insgesamt 263 Einsprungadressen für Systemroutinen zur Verfügung. Diese werden beim System-Reset ins RAM kopiert und sind auch nur dort funktionsfähig. Deshalb wurde auch die Dekodierung nicht im unteren ROM vorgenommen, sondern im RAM, wo sie tatsächlich benutzt werden.

# Das RAM des CPC464

| Address | Usage |
|---|---|
| FFFF | |
| C000 | Screen Memory |
| BF00 | System Stack |
| BE00 | ? |
| BDCD | Firmware Indirections |
| BDA3 | Entries to Integer Arithmetics |
| BD3D | Entries to Real Arithmetics |
| BD13 | Entries to the Machine Pack |
| BCC8 | Entries to the Kernel |
| BCA7 | Entries to the Sound Manager |
| BC65 | Entries to the Cassette Manager |
| BBFF | Entries to the Screen Pack |
| BBBA | Entries to the Graphics VDU |
| BB4E | Entries to the Text VDU |
| BB00 | Entries to the Key Manager |
| B900 | Entries to the High Kernel |
| B8E4 | Store for Arithmetics |
| B8DC | Store for the Line Editor |
| B800 | Store for Cassette Data |
| B550 | Store for Sound Data |
| B34C | Store for Keyboard Data |
| B328 | Store for Graphics VDU |
| B20C | Store for Text VDU |
| B1C8 | Store for System Screen Data |
| B100 | Store for Kernel Data |
| B0C2 | Basic FAC |
| AE8B | Basic Stack |
| ADB2 | Basic Sound Store |
| AC44 | Basic Edit Buffer |
| AC1C | Basic Store |
| AC00 | Basic Indirections |
| AB80 ?? | User Symbols (After 240.) |
| ???? | Memory Pool used for:<br>User Symbols<br>I/O Buffers<br>String Data<br>Indexed Variables<br>Variables |
| 0170 | Basic Program |
| 0040 | used by the Basic Line Assembler |
| 0000 | Restart and System entries |

## Zugriff auf ROM und RAM

Eines der wichtigsten Dinge, über die man sich als Programmierer im Klaren sein muß, ist die Art und Weise, wie man auf ROM oder RAM zugreift oder wie man die im ROM oder RAM befindlichen Routinen nutzt. Mit PEEK und POKE im Basic treffen Sie immer nur ins RAM. Wenn Sie aber eigene Maschinenroutinen einsetzen wollen, haben Sie es mit ROM-Select, ROM-Disable und ROM-Enable zu tun.

Grundsätzlich gilt fogendes: Schreibzugriffe beziehen sich immer auf's RAM, gleichgültig, welches ROM selectiert oder enabled wurde. Bei allen Lesezugriffen, hierzu zählt auch die Ausführung eines im ROM oder RAM gelegenen Programms, hat das ROM Vorrang, sofern es vorhanden und enabled ist. Der mittlere RAM-Bereich von 4000 bis BFFF, der ja von keinem ROM überlagert ist, kann jedoch immer beschrieben oder gelesen werden, unabhängig von der ROM-Selection. Das ist auch der Grund, warum sich die Einsprungadressen für die Betriebssystemroutinen, die ROM-Selections-Logik, die Interrrupt-Behandlung, die Bildschirm-Ausgabe, sämtliche Basic- und Betriebssystem-Parameter wie Tastaturbelegung oder Tastaturpuffer, Ein-/Ausgabepuffer, in diesem Bereich befinden. (Siehe gelbe Seiten.)

Beachten Sie, daß sich ROM-Select immer nur auf das obere ROM (ab C000) bezieht, während ROM-Enable/Disable sowohl das untere als auch das obere ROM unabhängig voneinander ein- oder ausschalten kann. Lassen Sie sich nicht dadurch verwirren, daß ROM-Select und ROM-Enable/Disable im gleichen Byte oder Register übergeben werden. Werte von 00 bis FB wählen das entsprechende (obere) ROM, enablen es und disablen das untere ROM. (Man verzeihe mir, besonders hier, mein schlechtes Deutsch!) Bei Werten von FC bis FF ändert sich nichts an der ROM-Selection, vielmehr werden durch die Bits 0 und 1 des Wertes die ROMs disabled oder enabled. 1B0 disables the lower ROM and 1B1 disables the upper ROM (so!).

Gehen wir davon aus, daß Sie den Bildschirm-Speicher dort lassen wo er ist (ab C000) und Ihr Basic-Programm mit verschiedenen Maschinenroutinen verbessern oder erweitern wollen. Solange Sie alles selbst strikken, also keine Routinen aus dem Betriebssystem oder dem Basic einbauen wollen, brauchen Sie nichts zu berücksichtigen. Spätestens dann, wenn Sie Vorhandenes nutzen wollen (und deshalb haben Sie vielleicht auch dieses Buch gekauft), müssen Sie:

- Daten, die Sie bearbeiten wollen, in den mittleren RAM-Bereich von 4000 bis BFFF bringen,
- Betriebssystem-Routinen (damit ist das untere ROM gemeint) nur über die vorhandene Sprungliste ansprechen,
- Routinen in einem oberen ROM (z. B. Basic) mit einem entsprechenden RST 3 ansprechen.

Studieren Sie am besten die Basic-Routinen, die mit Ihrem Problem verwandt sind. Wenn Sie voll in die Programmierung einsteigen wollen, werden Sie nicht um die Anschaffung des Firmware-Handbuches und des DEVPAC Assembler/Disassembler's herumkommen. Für die ersten Versuche kommen Sie vielleicht mit den im Anhang abgedruckten Tabellen zurecht.

## Indirection — Umleitung

All das, was man früher bei anderen Rechnern, wenn es überhaupt möglich war, mit viel List und Tücke aufpfropfen mußte, ist hier schon vorgesehen: Für das Basic sind insgesamt neun verschiedene Umleitungen eingebaut. Anstelle des dort normalerweise enthaltenen Codes C9 (ret) kann ein Jump zu einer Anwenderroutine eingesetzt werden. Dies ist besonders zur Fehlerbehandlung oder zur Befehlssatzerweiterung nützlich. Man muß davon ausgehen, daß die künftig lieferbaren Hardware-Erweiterungen, wie z. B. ein Floppy-Laufwerk, von dieser Möglichkeit ausgiebig Gebrauch machen werden.

## Multitasking

Ein Begriff, unter dem man sich schwer etwas Konkretes vorstellen kann. Der CPC läßt keine Wünsche offen: Es geht sogar mit Basic. Dem Assembler-Programmierer stehen Sytemroutinen zur Verfügung, mit denen man eigene Tasks 'ein- und ausklinken' kann.

## Das Basic

Das Basic besteht im wesentlichen aus den folgenden Teilen:

### 1. Der Programmeditor

Er bildet die direkte Schnittstelle zum Anwender. Alles, was über die Tastatur eingegeben wird, wird zunächst im Editbuffer gespeichert. Erst wenn die Eingabe (mit RETURN) abgeschlossen ist, übernimmt der Line-Assembler die weitere Ausführung.

### 2. Der Line-Assembler

hat die Aufgabe, den im Edit-Buffer abgelegten Text in das Basic-Format zu komprimieren, d. h. Basic-Befehle (Schlüsselwörter) werden in das entsprechende Token umgewandelt. Variable werden je nach Typ gekennzeichnet und Konstantwerte in ein genormtes Format umgewandelt. Der CPC464 verwendet zur Zwischenspeicherung den Bereich ab 0040 unterhalb des Basic-Programms. Begann die Eingabe mit einer Zeilennummer, so wird diese Progammzeile in das Basic-Programm eingebaut. Fehlt die Zeilennummer, so werden die Statements unmittelbar vom Interpreter ausgeführt.

### 3. Der Basic-Interpreter

Dieser interpretiert - freilich nach festen Regeln - den kompakten Basic-Code und führt die dort enthaltenen Kommandos und Funktionen durch Aufrufen der entsprechenden Interpreter-Routinen aus.

### 4. Der List-Generator

Schließlich ist es nicht damit getan, ein Programm einzutippen und kompakt abzuspeichern. Genauso wichtig ist es, diese Kompaktform wieder lesbar zu machen, sei es, um sie zeilenweise an den Editor zu übergeben für Änderungen oder um als Listing auf Bildschirm oder Drucker ausgegeben zu werden.

## Utilities

sind Routinen, die mehr oder weniger notwendig oder nützlich sind: Hier gibt es von Basic zu Basic die größten Unterschiede. Doch sind es gerade diese Feinheiten, die das Arbeiten in Basic zum Vergnügen oder zur Qual machen können. Der CPC464 kann sich hier sehen lassen: Alles was das Herz begehrt, Chain Merge mit Delete ‹Zeilenbereich› und ‹Startzeile›, Sound, Windows, Graphic, Key Def, Symbol, Instr, Midstr() = , ja und Multitasking!

## Das Basic des CPC464

Was ist also nun das Besondere, das Herausragende an diesem Basic? Zum einen ist es der erhebliche Umfang der Funktionen und Befehle, zum anderen ist es seine Geschwindigkeit. Wenn bei einem Benchmark-Test der CPC erheblich schneller arbeitet als vergleichbare Produkte, so kann dies verschiedene Ursachen haben: Der Prozessortyp, die Taktfrequenz und die Raffinesse des Programms. Ich will hier nicht 6502 oder 6510 gegen Z80 abwägen, doch seit ich mich mit dem Z80 befasse, ist er mir allein schon von der Anzahl und Art der Register her sowie der Vielfalt der zur Verfügung stehenden Befehle sympatischer. Doch nicht der Prozessor, sondern die Software ist hier das Entscheidende.

Wenn wir auf die oben erwähnte Unterteilung der verschiedenen Basic-Funktionen zurückkommen, so befindet sich dort nur ein Abschnitt, der zeitkritisch ist: Der Interpreter. Nicht ohne Grund steht Basic in dem (Ver-)Ruf, ca. 20 bis 1000 mal langsamer zu sein als ein vergleichbares Maschinenprogramm. Die Begründung liegt auf der Hand: Weil der Interpreter die Programmzeile ja erst einmal umwandeln muß, um zu sehen, was er womit machen soll. Je leichter man also dem Interpreter das Erkennen seiner Aufgabe macht, desto schneller wird er schließlich, und genau hier setzt die Raffinesse des CPC-Basic ein. Der Interpreter wurde (fast) vollständig von allen zeitaufwendigen Routinen, wie z.B. Zahlenumwandlung, Variablen suchen usw. entlastet.

Ein Byte kann, wie wir wissen, Werte von 0 bis 255 oder hex 00 bis FF annehmen. Doch erst die Vereinbarung über die Bedeutung des Inhalts gibt dem Byte seinen Sinn. Im Anhang sind die 255 Bytes mit ihren möglichen Bedeutungen für den CPC464 aufgelistet.

Eine der angenehmen Eigenschaften des CPC ist, daß er intern wie extern den Ascii-Code zur Textdarstellung verwendet. Das erspart schon einmal zeitraubende Umwandlungen bei der Ein- und Ausgabe und ermöglicht den problemlosen Anschluß von Druckern mit Centronics-Schnittstelle. Doch hier gleich einen Wermutstropfen vorneweg: die eingebaute Druckerschnittstelle liefert nur sieben Datenbits, das achte wurde als Strobe verwendet. Für den Normalbetrieb spielt das zwar keine Rolle, doch wenn man Grafiken ausdrucken möchte fehlt es einfach.

Wenn man die Ascii-Tabelle betrachtet, so stellt man fest, daß nur sieben Bit verwendet werden. Das achte Bit wird, neben der eben erwähnten Graphic-Steuerung von Druckern, gelegentlich als Parity-Bit verwendet, was aber nur bei der bit-seriellen Datenübertragung, möglicherweise über Telefonleitung, eine Rolle spielt. Bei den heute üblicherweise kurzen

und sicheren Datenübertragungswegen vom Rechner zum Drucker oder Bildschirm-Terminal bleibt es jedoch meist unberücksichtigt. (Außerdem muß man es extra erzeugen und auch noch prüfen. Dieser Hard- oder Software-Aufwand wird gerne eingespart.) Von den unterhalb von hex 20 liegenden Kontrollzeichen sind die bekanntesten CR, LF, BS, TAB und ESC. Für den Basic-Interpreter haben jedoch die Zeichen von 00 bis 1F eine völlig andere Bedeutung. Wegen der Wichtigkeit für das Verständnis des Aufbaus einer Basic-Zeile sei hier die Tabelle gleich eingefügt.

```
00  <line end>                10  <const 2>
01  <statement end>           11  <const 3>
02  <integer VAR%>            12  <const 4>
03  <string VAR$>             13  <const 5>
04  <real VAR!>               14  <const 6>
05  <FAC real>                15  <const 7>
06                            16  <const 8>
07                            17  <const 9>
08                            18
09  <allow for spaces>        19  <next 1 byte VAL>
0A                            1A  <next 2 byte VAL>
0B  <integer var>             1B  <next 2 byte BIN>
0C  <string var>              1C  <next 2 byte HEX>
0D  <real var>                1D  <next 2 byte ADDRESS>
0E  <const 0>                 1E  <next 2 byte LINE#>
0F  <const 1>                 1F  <next 5 byte REAL>
```

```
10 REM Analyse
20 GOSUB 160
50 FOR iii=&40 TO HIMEM STEP 16
60 MID$(pri,1,77)=SPACE$(77):MID$(pri,1,4)=HEX$(iii,4)
70 FOR jjj=0 TO 15
80 kkk=PEEK(iii+jjj):mmm=kkk AND 127:IF mmm<&20 OR mmm=127 THEN mmm=46:'.
90 MID$(pri,(56+jjj),1)=CHR$(mmm)
100 MID$(pri,(jjj*3+1)+6,4)=FNxxx(kkk):NEXT jjj
110 IF MID$(lpri$,5)<>MID$(pri,5) GOTO 140
120 IF flag%=0 THEN PRINT#ch,"       ***":flag%=1:GOTO 150 ELSE 150
140 MID$(lpri$,1,77)=pri:PRINT#ch,pri:flag%=0
150 NEXT:CLOSEOUT:END
160 DEFINT c,l-m:DEFSTR p,s-z:MODE 2:ch=8
170 DIM array(3,4):array(3,4)=SQR(PI):xlotto!=13983816
180 DEF FN xxx(jj)=HEX$(jj,2)
185 pri=SPACE$(77):lpri$=pri
190 RETURN
```

Dies soll ein Programm sein, das einerseits den gesamten Speicherbereich des CP464 ausdruckt, zum anderen sollten möglichst viele Commandotypen und Variablenarten verwendet werden, um deren Aufbau zu beschreiben. Wundern Sie sich auch bitte nicht über die etwas ungewöhnliche Art der Zuweisung »MIDstr(pri,1,77) = SPACEstr(77)«, diese soll nur verhindern, daß der Speicher mit immer neuen Strings belegt wird (und natürlich findet auch kein Garbage Collect statt). Die Variablennamen wurden so gewählt, daß sie auch im Hexdump deutlich zu erkennen sind. Soll die Ausgabe auf den Bildschirm erfolgen, ändern Sie in Zeile 160: ch = 0. Die Zeile 170 ist überflüssig.

Wenn Sie bei eigenen Versuchen das Programm ändern, laufen lassen und wieder ändern, sieht der Speicher nicht mehr so aufgeräumt aus. Sie sollten dann das Programm sichern, den Rechner zurücksetzen und das Programm von der Cassette starten.

So wurde es auch hier gemacht. Das erkennt man gleich ab Adresse 0040. Dort legt der Line-Assembler eine Basic-Zeile ab, bevor diese ins Programm eingebaut oder als Direct Command gleich ausgeführt wird. CA ist das Token für RUN, dann folgt das Anführungszeichen und der Programmname. Haben Sie das Programm geändert, so steht die zuletzt geänderte Zeile in diesem Bereich, allerdings sind die ersten Bytes immer mit CA 00 00 00 00 überschrieben, wenn sie das Programm starten.

Betrachten wir nun den Speicherbereich ab 0170. Eine Programmzeile ist nach folgendem Schema aufgebaut:

‹Länge› ‹Zeilennummer› ‹Statement(s)› ‹Endezeichen›

Die Länge besteht aus zwei Bytes und dient zur Errechnung der Adresse der nächsten Zeile. Die Zeilennummer wird ebenfalls mit zwei Bytes dargestellt (immer niederwertiges Byte zuerst). Ein Statement versteht sich einschließlich der eventuell erforderlichen Parameter. Mehrere Statements in einer Zeile sind durch ein Trennzeichen (:) 01 voneinander getrennt. Das Ende einer Basic-Zeile wird intern duch 00 markiert. Das Ende des Basic-Programms wird durch eine neue Zeile mit der Länge 0000 gekennzeichnet. Die Zuweisung an eine Variable nimmt eine Sonderstellung ein, weil es das einzige Statement ist, das nicht mit einem Token beginnt (wenngleich LET aus Kompatibilitätsgründen zugelassen ist).

Unsere erste Zeile hat die Länge 000E und die Zeilennummer 000A (10.). Zum leichteren Auffinden der Zeilenanfänge sind diese im Hexdump unterstrichen. Als erstes Token erscheint nun C5 für REM und anschließend der Text dazu, abgeschlossen durch 00.

```
0040  CA 22 64 65 6D 6F 00 00 00 00 00 00 00 00 00 00   J"demo..........
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
      ***
0170  0E 00 0A 00 C5 20 41 6E 61 6C 79 73 65 00 0A 00   ....E Analyse...
0180  14 00 9F 20 1D 43 03 00 1B 00 32 00 9E 20 0D 37   ... .C....2.. .7
0190  00 69 69 E9 EF 1C 40 00 20 EC 20 FF 42 20 E6 20   .iiio.@. l .B f
01A0  19 10 00 35 00 3C 00 AC 28 0C 24 00 70 72 E9 2C   ...5.<.,(.$.pri,
01B0  0F 2C 19 4D 29 EF FF 16 28 19 4D 29 01 AC 28 0C   .,.M)o..(.M).,(.
01C0  24 00 70 72 E9 2C 0F 2C 12 29 EF FF 73 28 0D 37   $.pri,.,.)o.s(.7
01D0  00 69 69 E9 2C 12 29 00 14 00 46 00 9E 20 0D 42   .iii,.)...F.. .B
01E0  00 6A 6A EA EF 0E 20 EC 20 19 0F 00 59 00 50 00   .jjjo. l ...Y.P.
01F0  0D 4D 00 6B 6B EB EF FF 12 28 0D 37 00 69 69 E9   .M.kkko..(.7.iii
0200  F4 0D 42 00 6A 6A EA 29 01 0B 58 00 6D 6D ED EF   t.B.jjj)..X.mmmo
0210  0D 4D 00 6B 6B EB 20 FA 20 19 7F 01 A1 20 0B 58   .M.kkk z ...! .X
0220  00 6D 6D ED F1 1C 20 00 20 FC 20 0B 58 00 6D 6D   .mmmq. . | .X.mm
0230  ED EF 19 7F 20 EB 20 0B 58 00 6D 6D ED EF 19 2E   mo.. k .X.mmmo..
0240  01 01 C0 2E 00 27 00 5A 00 AC 28 0C 24 00 70 72   ..@..'.Z.,(.$.pr
0250  E9 2C 28 19 38 F4 0D 42 00 6A 6A EA 29 2C 0F 29   i,(.8t.B.jjj),.)
0260  EF FF 03 28 0B 58 00 6D 6D ED 29 00 38 00 64 00   o..(.X.mmm).8.d.
0270  AC 28 0C 24 00 70 72 E9 2C 28 0D 42 00 6A 6A EA   ,(.$.pri,(.B.jjj
0280  F6 11 F4 0F 29 F4 14 2C 12 29 EF E4 0C 1C 00 78   v.t.)t.,.)od...x
0290  78 F8 28 0D 4D 00 6B 6B EB 29 01 B0 20 0D 42 00   xx(.M.kkk).0 .B.
02A0  6A 6A EA 00 25 00 6E 00 A1 20 AC 28 03 2E 00 6C   jjj.%.n.! ,(...l
02B0  70 72 E9 2C 13 29 F2 AC 28 0C 24 00 70 72 E9 2C   pri,.)r,(.$.pri,
02C0  13 29 20 A0 20 1D 05 03 00 3D 00 78 00 A1 20 02   .)   ....=.x.! .
02D0  61 00 66 6C 61 E7 EF 0E 20 EB 20 BF 23 0B 06 00   a.flago. k ?#...
02E0  63 E8 2C 22 20 20 20 20 20 20 2A 2A 2A 22 01 02   ch,"      ***"..
02F0  61 00 66 6C 61 E7 EF 0F 01 A0 20 1D 39 03 20 01   a.flago..  .9. .
0300  97 20 1D 39 03 00 34 00 8C 00 AC 28 03 2E 00 6C   . .9..4...,(...l
0310  70 72 E9 2C 0F 2C 19 4D 29 EF 0C 24 00 70 72 E9   pri,.,.M)o.$.pri
0320  01 BF 23 0B 06 00 63 E8 2C 0C 24 00 70 72 E9 01   .?#...ch,.$.pri.
0330  02 61 00 66 6C 61 E7 EF 0E 00 0A 00 96 00 B0 01   .a.flago......0.
0340  89 01 98 00 20 00 A0 00 8E 20 63 2C 6C 2D 6D 01   .... . .. c,l-m.
0350  90 20 70 2C 73 2D 7A 01 AD 20 10 01 0B 06 00 63   . p,s-z.- .....c
0360  E8 EF 16 00 3A 00 AA 00 93 20 0D 00 00 61 72 72   ho..:.*.. ...arr
0370  61 F9 28 11 2C 12 29 01 0D 0B 00 61 72 72 61 F9   ay(.,.)....array
0380  28 11 2C 12 29 EF FF 18 28 FF 44 29 01 04 11 00   (.,.)o..(.D)....
0390  78 6C 6F 74 74 EF EF 1F 00 48 60 55 98 00 22 00   xlottoo..H`U..".
03A0  B4 00 8D 20 E4 20 0C 1C 00 78 78 F8 28 0D 00 00   4.. d ...xxx(...
03B0  6A EA 29 EF FF 73 28 0D 00 00 6A EA 2C 10 29 00   jj)o.s(...jj,.).
03C0  21 00 B9 00 0C 24 00 70 72 E9 EF FF 16 28 19 4D   !.9..$.prio..(.M
03D0  29 01 03 2E 00 6C 70 72 E9 EF 0C 24 00 70 72 E9   )....lprio.$.pri
03E0  00 06 00 BE 00 C9 00 00 00 00 00 43 C8 01 08 00   ...>.I.....CH...
03F0  00 00 58 4C 4F 54 54 CF 04 00 48 60 55 98 00 00   ..XLOTTO..H`U...
0400  58 58 D8 42 AC 03 00 00 50 52 C9 02 4D 2F A6 00   XXXB,...PRI.M/&.
0410  00 4C 50 52 C9 02 4D E2 A5 00 00 49 49 C9 04 00   .LPRI.Mb%..III..
0420  00 00 04 8B 00 00 4A 4A CA 04 00 00 00 50 84 00   ......JJJ....P..
0430  00 4B 4B CB 04 00 00 00 00 00 00 00 4D 4D CD 01   .KKK........MMM.
0440  2E 00 00 00 46 4C 41 C7 01 00 00 00 00 41 52 52   ....FLAG.....ARR
0450  41 D9 04 69 00 02 05 00 04 00 00 00 00 00 00 00   AY.i............
0460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
      ***
04B0  00 00 00 00 00 00 00 00 00 8E C4 DF 62 81 00 00   ..........D_b...
04C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
      ***
```

```
       ***
A550   20 20 30 30 20 30 30 20 30 30 20 30 30 20 30 30     00 00 00 00 00
A560   20 33 30 20 33 30 20 32 30 20 33 30 20 33 30 20    30 30 20 30 30
A570   33 33 20 33 30 20 32 30 20 33 33 20 33 30 20 32   33 30 20 33 30 2
A580   32 20 20 33 33 20 33 30 20 32 30 20 33 33 20 33   2  33 30 20 33 3
A590   33 20 33 20 20 20 20 20 20 20 20 20 20 20 20 20   3 3
A5A0   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
       ***
A5D0   20 20 20 20 20 20 20 20 20 20 20 20 20 20 41 35                 A5
A5E0   45 35 41 35 44 30 20 20 32 30 20 32 30 20 32 30   E5A5D0   20 20 20
A5F0   20 33 35 20 34 34 20 33 30 20 32 30 20 32 30 20    35 44 30 20 20
A600   33 30 20 32 30 20 33 32 20 33 30 20 32 30 20 33   30 20 32 30 20 3
A610   30 20 32 30 20 33 33 20 20 33 30 20 32 30 20 33   0 20 33   30 20 3
A620   20 20 33 30 20 32 30 20 33 20 20 20 20 20 20 41     30 20 3        A
A630   36 33 30 20 20 33 36 20 33 33 20 33 30 20 32 30   630   36 33 30 20
A640   20 20 20 20 20 20 20 32 30 20 33 32 20 33 30 20           20 32 30
A650   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 32                  2
A660   20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
A670   20 20 20 20 20 20 20 20 20 20 20 20
```

Die zweite Zeile hat die Länge 000A und die Nummer 0014 (20.). Dann folgt das Token 9F für GOSUB und ein Blank. Was nun folgt, unterscheidet den CPC464 wesentlich von anderen Basic-Rechnern. Eine Zeilennumer 160 ist hier nicht zu finden. Stattdessen hat der Rechner (beim ersten Durchgang heimlich) die Zeilennummer gegen die tatsächliche Speicheradresse ausgetauscht und muß nun künftig nie mehr nach dieser Nummer suchen. Gleichzeitig wurde auch das Kennzeichen 1E (Zeilennummer) in 1D (Adresse) geändert.

Die dritte Zeile lautet: FOR iii = &40 .... Entsprechend finden wir das Token 9E und Blank vor, doch bevor es mit 'iii' weitergeht, sind da noch drei Bytes: 0D als Kennzeichen für eine (vordefinierte) Real-Variable und 0037 als direkter Verweis in die Variablentabelle (deren Beginn wir mit PEEK(&AE85) + PEEK(&AE86)*256 erfragen können). Spätestens jetzt wird das Prinzip klar, nach dem hier gearbeitet wird: Vermeidung aller zeitaufwendigen Arbeiten durch direkten Verweis auf die jeweiligen Speicherstellen. Zwar muß die Variable bei Ihrer ersten Verwendung im Programm in die Tabelle eingetragen werden, denn beim Programmstart stehen anstelle dieses Verweises nur zwei Nullen als Platzhalter, doch spätestens beim zweiten Durchgang kann unmittelbar auf den Inhalt zugegriffen werden. Taucht eine Variable im Programmtext auf, so wird nur geprüft, ob der Platzhalter bereits durch einen Tabellenindex ersetzt wurde. Nur wenn eine Variable an dieser Stelle für das Programm neu ist, wird sie in der Variablentabelle gesucht und der ensprechende Index ins Programm eingebaut.

Nun folgt der Variablen-Namen. Er muß auf jeden Fall mit einem Buchstaben beginnen, darf bis zu 40 Zeichen lang sein und außer Buchstaben und Ziffern auch Punkte enthalten. Dies ist eine weitere angenehme Eigenschaft des CPC-Basic, die es ermöglicht, durch Verwendung selbsterklärender Variablen-Namen (z. B. lines.to.print, lines.printed, page.number), übersichtliche Programme zu schreiben.

Das Ende des Variablen-Namens wird durch das gesetzte Bit 7 gekennzeichnet. Besteht der Name nur aus einem Zeichen, ist es bereits dort gesetzt und könnte, wenn Sie mit dem oben angeführten kleinen Basic-Programm arbeiten, beim schnellen Hinsehen auch mit einem Token verwechselt werden. Verwenden Sie daher für Ihre ersten Experimente am besten Variablen-Namen aus mehreren gleichen Buchstaben, die Sie dann auch in hex-Form leicht erkennen können.

Auf den Variablen-Namen folgt in dem oben beschriebenen Beispiel der Zuweisung das Token für ' = '. Dieses Token weist den Interpreter an, den Wert des nun folgenden Ausdrucks an die Variable (deren Platz innerhalb der Tabelle ja bereits bekannt ist) zuzuweisen. In unserem Beispiel wird einfach der Wert einer Zahl zugewiesen. (Dies gilt allgemein, bei der Zuweisung an eine Laufvariable finden noch andere Vorgänge statt, die hier nicht beschrieben sind.) Was nun folgt, sind nicht einfach eine Reihe von Ziffern im Ascii-Format, nein, auch hier herrscht Raffinesse, denn die Arbeit der Umsetzung hat der Line-Assembler bereits längst erledigt und serviert das ganze mundgerecht. Das nächste Byte sagt genau was kommt: Eine weitere Variable oder eine Konstante mit genauer Formatangabe. Für die Variablen gilt die gleiche Vereinbarung wie oben bereits beschrieben. Häufig verwendete Konstante im Bereich von 0 bis 9 sind bereits in diesem Byte selbst verschlüsselt, nämlich dann, wenn dieses einen Wert von 0E ( = 0) bis 17 ( = 9) einnimmt. Das spart Speicherplatz, und nach einfacher Subtraktion von 0E steht der Wert zur Weiterverarbeitung bereits zur Verfügung.

19 gibt an, daß das nun folgende Byte die Zahl darstellt, 1A, daß die Zahl aus den folgenden zwei Bytes (L,H) besteht, ebenso wie bei 1B und 1C, nur mit dem Unterschied, daß die Zahl im Binär- bzw. Hexadezimal-Format eingegeben wurde (und natürlich auch bei der Ausgabe wieder so dargestellt wird). 1D erklärt die folgenden zwei Bytes eine Adresse sind, 1E sagt aus, daß die folgenden zwei Bytes eine Zeilennummer darstellen, und 1F schließlich verweist auf die nun folgenden fünf Bytes im Real-Format.

Aber nicht nur die aufbereitete Zahlendarstellung bringt Vorteile, mehr noch deren Verarbeitung. Wird z. B. einer Integer-Variablen ein Integer-Wert zugewiesen, so geschieht dies schlicht durch Kopieren von zwei Bytes. Dasselbe gilt für die fünf Bytes bei der Zuweisung einer Real-Zahl an eine Real-Variable oder den drei Bytes bei einer einfachen String-Zuweisung. Stimmt der Variablen-Typ bei Real und Integer nicht überein, so wird zusätzlich intern eine Umwandlung vorgenommen.

All diese Maßnahmen wirken sich letztlich in einer sehr hohen Verarbeitungsgeschwindigkeit aus, Vorverlegung der zeitintensiven Zahlenumwandlungen in die Programmerstellungsphase, direkter Verweis auf den Variablen-Tabellen-Eintrag sowie Vorgabe des Verarbeitungsschlüssels.

In der Basic-Zeile folgt nun entweder 00 als Zeichen für das Zeilenende oder 01 als Zeichen für den Doppelpunkt, was heißt, daß weitere Statements folgen können. Bleibt anzumerken: Gleichgültig, ob Sie nun mit vordefinierten Variablen-Typen (DEFINT, DEFSTR, DEFREAL) arbeiten oder ob Sie an den Variablen-Namen '%' oder '!' anhängen, ob Sie Integer-Zahlen als Dezimal-Ziffern hex oder binär eingeben, Sie belegen immer den gleich viel Speicherplatz, nur optisch beim Auflisten der Programme belegen die Zeichen '%, !, &' eine Druckstelle.

Etwas Gutes hat das Anhängsel aber doch, wenn Sie mit Bytes geizen müssen (und es nicht auf die Lesbarkeit des Programms ankommt): Das Anhänsel ersetzt einen Blank nach dem Variablennamen und spart bei bestimmten Gelegenheiten ein Byte ein. Beispiele:

```
PRINT a!i%b!c$
IF a%OR b%OR i%THEN 300
IF a%AND&7F GOTO 300
```

Doch Vorsicht: Eine vordefinierte Real-Variable, z.B. i ist manchmal auch gleich i%. Zur Veranschaulichung geben Sie bitte folgendes ein:

```
NEW
i=1 : i%=2 : ? i , i%
DEFINT i : ? i , i%             'Merken Sie was?
```

Des Rätsels Lösung ist ganz einfach: DEFINT, DEFREAL und DEFSTR veranlassen keinerlei Umwandlung im Rechner, es sind nur die Anweisungen an den Interpreter, Variablennamen ohne Anhängsel als Variable des angegebenen Typs zu betrachten. Es ist gute Praxis, diese Definitionen an den Anfang des Programms zu stellen und einen bestimmten Buchstabenbereich, z. B. I-N (wie in Fortran) grundsätzlich als Integer zu deklarieren.

Auf den gelben Seiten in diesem Buch sehen wir bei den Adressen AE7B bis AE89 die verschiedenen Basic-Zeiger. Dabei fällt auf, daß für Programm-Ende (AE83) und Beginn der Variablentabelle (AE85) eigene Zeiger existieren, die zwar denselben Wert anzeigen, aber nicht anzeigen müssen. Das dürfte eine Möglichkeit eröffnen, von der Basic-Programmierer bislang nur träumten: Unterprogramme mit lokalen Variablen, unabhängig vom Hauptprogramm. Wohlgemerkt, das läuft so noch nicht, doch kann es bei entsprechender Basic-Erweiterung möglich werden.

In unserem Musterprogramm beginnt die Variablentabelle der einfachen Variablen bei 03E9. Sie ist wie folgt aufgebaut:

00 00 N A M E ‹Type› ‹Inhalt›

Bei 0000 handelt es sich wohl um eine Trennungsmarkierung für den Namen. Der NAME ist in Großbuchstaben dargestellt und hat im letzten Buchstaben das Bit 7 gesetzt. Im Gegensatz zum Programmtext gelten für ‹Type›:

| | |
|---|---|
| 01 | Integer |
| 02 | String |
| 04 | Real |

‹Type› + 1 ergibt die Länge des Inhalts, der sich unmittelbar anschließt (es sind nicht grundsätzlich 5 Bytes, wie bei anderen Interpretern). Die drei Bytes bei Strings bedeuten:

‹Länge› ‹Adresse›

Ist die Länge 00, so hat die Adresse keine Bedeutung. Der String PRI (Descriptor bei 040C) liegt bei A62F, also ganz oben. Der String LPRI (Descriptor bei 0416) liegt direkt unterhalb, bei A5E2. Von dem gesamten Stringbereich sind aber noch weitere 146 Bytes belegt. Das sind zweimal 73 Bytes aus dem Vergleich in Zeile 110. Man sieht, daß für diesen Vergleich zwei Strings angelegt, aber unmittelbar danach wieder freigegeben werden.

Auch Funktions-Namen sind in der Variablentabelle eingetragen. Zur Unterscheidung ist bei diesen im ‹Type› das Bit 6 gesetzt. Auf den Funktions-Typ folgt die Adresse, wo die Funktion im Programm definiert ist. In unserem Beispiel ist dies 03AC. Diese Adresse zeigt auf die '(', die an den Funktionsnamen im Programm anschließt. Zu beachten ist, daß hier eine Stringfunktion definiert wurde, was anderswo keineswegs selbstverständlich ist.

Die dimensionierten (oder indizierten) Variablen werden in einer eigenen Tabelle verwaltet. Der Zeiger auf den Anfang der Tabelle steht in AE87, in unserem Beispiel ist dies 044B. Namenseintrag und Variablentype stimmen mit den einfachen Variablen überein.

00 00
N A M E
‹Type›
‹Länge des Eintrags›
‹Anzahl der Dimensionen›
‹Anzahl Felder n.te Dimension›
‹Anzahl Felder (n-1)te Dimension
‹Anzahl Felder 1. Dimension›
‹Inhalt von (0,0)›
‹Inhalt von (0,1)› ... (n,0) (n,1) (n,2)

Die Länge des Eintrags errechnet sich Ausdehnung1 * Ausdehnung2 * Ausdehnung.n * Bytes pro Element + 5 (das sind die zwei Bytes vor dem Variablennamen, ein Byte für den Variablentyp und die zwei Bytes, in denen die Länge des Eintrags steht). Tatsächlich ist noch die Länge des Variablennamens zu addieren, will man den durch die Variable benötigten Speicherplatz errechnen.

## Fließkomma-Operationen

werden hauptsächlich in einem ganz bestimmten Speicherbereich, nämlich dem Fließkomma-Akkumulator (FAC) ausgeführt. Fließkommazahlen werden in einem besonderen Format in fünf aufeinanderfolgenden Bytes abgespeichert. Da der FAC aber auch für Integer-Zahlen oder String-Zeiger verwendet wird, ist dem FAC ein Byte vorangestellt, das den augenblicklich im FAC enthaltenen Variablen-Typ kennzeichnet. Diese Speicherstelle ist als VARTYPE bezeichnet. Die möglichen Werte für VARTYPE sind: 02 für Integer, 03 für String, 05 für Real (Fließkomma). Dieser Schlüssel gibt gleichzeitig auch die Anzahl der für eine Operation benötigten Bytes an. Nachstehendes Schema soll dies verdeutlichen.

|  | Real | Integer | String |
|---|---|---|---|
| FAC + 4: | Sign, Exponent |  |  |
| FAC + 3: | Sign, Mantissa |  |  |
| FAC + 2: | Mantissa |  | Address H |
| FAC + 1: | Mantissa | H | Address L |
| FAC: | Mantissa, LSB | L | Length |
| VARTYPE: | 05 | 02 | 03 |

Das Betriebssystem verwendet an anderer Stelle weitere Fließkomma-Akkumulatoren FAC1, FAC2, FAC3 als Zwischenspeicher bei arithmetischen Routinen.

Die arithmetischen Routinen des Systems sind über eine Sprungliste im Bereich BD3D bis BDCA anzusprechen. Für Integer- und Real-Berechnungen stehen jeweils eigene Routinen zur Verfügung. Die Parameter werden in Registern übergeben, bei Real-Berechnungen sind die Register Zeiger auf die Rechen- oder Ergebnis-Variablen, bei den Integer-Routinen werden die Werte in den Registern direkt übergeben.

Beispiel für den Aufruf einer Real-Berechnung:

```
var1:     equ 8000        ;any 5 locations above 4000
var2:     equ 8005        ;any 5 locations above 4000
mult:     equ BD61        ;Aufruf über Sprungliste
          ld hl,var1      ;pointer to the contents
          ld de,var2      ;pointer to the contents
          call mult
                          ;(var1) contains the product
```

Beispiel für eine Integer-Berechnung:

```
calcmin:  ld hl,(secu)    ;hole Sekundenwert
          ld de,3C        ; = 60.
          call BDB8       ;integer division hl = hl/de
          ld (minu),hl    ;speichere Ergebnis
```

## Hinweis zur Tastatur

Wenn Sie das laute Klappern der Leertaste stört, umwickeln Sie den darunterliegenden Drahtbügel mit ein bis zwei Lagen Isolierband, das hilft. (Leider muß man den Rechner dazu zerlegen.)

## Hinweis zum Cassettenrecorder

Es sind bis jetzt keinerlei Lesefehler aufgetreten, obwohl alles mit der Schreibgeschwindigkeit 2 aufgenommen wurde. Es empfiehlt sich, darauf zu achten, nach dem Einlesen oder Abspeichern die Stopptaste zu drücken, um Verformungen der Andruckwalze zu verhindern.

## Hinweis zum Basic-Handbuch

Es exisitiert eine Basic-Funktion, die nicht im Handbuch beschrieben ist. Vielleicht wurde sie einfach vergessen. Sie lautet:

@ ‹variablenname›

FUNKTION: Gibt die Speicheradresse zurück, an der sich der Tabelleneintrag von ‹variablenname› befindet.

Beispiel:

```
i = 0          ;Variable muß verwendet worden sein
PRINT @ i      ;Adresse des Tabelleneintrags
```

Bei indizierten Variablen erhält man die Adresse des angegebenen Elements, bei String-Variablen die Adresse des String-Descriptors, nicht des Strings. Der Descriptor besteht aus drei Bytes: String-Länge und String-Adresse. Durch Auswertung der String-Adresse kann auch auf den String-Inhalt zugegriffen werden. Doch Vorsicht: Nach einem Garbage-Collect muß die aktuelle Adresse neu ermittelt werden.

## Unbekannte Funktion?

Es existiert eine Basic-Funktion, von der ich weder weiß, wie man sie richtig aufruft, noch wozu sie eigentlich dient (F8EA):

DEC$(‹num expr›,‹string expr›)

Leider kam ich bei meinen Experimenten nicht über 'Syntax error' hinaus, obwohl laut Listing auch 'Improper argument' vorgesehen ist.

## Funktion des Disassemblers

Zum besseren Verständnis des nun folgenden Listings ist es von Vorteil, etwas über die Funktionsweise des zur Erstellung des Listings verwendeten Disassemblers zu wissen. Dieser verwendet im wesentlichen zwei Eingangsdateien. Zum ersten einmal die Daten über den ROM- oder RAM-Inhalt des CPC464, zum zweiten eine Kommandodatei, die dem Disassembler sagt, wie er die Speicherinhalte des CPC464 zu interpretieren hat, an welcher Stelle welche Überschriften oder Kommentare einzufügen sind, wo eine neue Seite zu beginnen ist, welcher Untertitel auf der Seite auszudrucken ist und natürlich auch, welche von vier verschiedenen Tabellen zur Interpretation von Byte-Werten heranzuziehen ist. Auf diese Kommandodatei wird sowohl sequentiell aufsteigend entsprechend der gerade bearbeiteten Adresse als auch wahlfrei bei einem Verweis auf eine Unterroutine oder Speicherstelle irgendwo im CPC-Programm zugegriffen. Auf Grund dieser Arbeitsweise werden sofort die Vorteile als auch mögliche Nachteile des Verfahrens deutlich. Vorweg die Nachteile: Wurde z.B. wegen menschlichen Versagens die an einer bestimmten Stelle eigentlich zu verwendende Tabelle nicht richtig angegeben, so erscheint natürlich auch im Listing ein falscher Kommentar. Dem findigen Leser wird es aber sicher nicht schwerfallen, anhand der im Anhang aufgeführten Tabelle die für dieses Byte zutreffende Interpretation zu finden. In dieser kann man auch gleich feststellen, wodurch sich die einzelnen Tabellen voneinander unterscheiden.

Dasgleiche trifft natürlich auch für die Kolumnentitel zu; solange kein neuer angegeben wurde, gilt immer noch der alte. Am gemeinsten sind jedoch die Einzelkommentare: Hat man sich da vertan, erscheint er entweder überhaupt nicht (weil keine Adresse paßt) oder an einer Stelle, an der er nichts verloren hat.

Doch nun zu den Vorteilen: Der gravierendste Vorteil wird an einem Beispiel deutlich: Wurde eine Speicherstelle oder ein Unterprogramm einmal mit einem Namen versehen, so wird überall dort, wo der Disassembler auf diese Speicherstelle zugreift oder das Unterprogramm aufruft, automatisch dieser Name in der Kommentarzeile eingefügt. Es versteht sich eigentlich von selbst, daß diese Kommentare nicht einen Assembler-Befehl kommentieren, sondern das, was dieser Befehl bewirkt. Der Kommentar ergibt nur zusammen mit dem Assembler-Befehl seinen Sinn. Das mag bei 'CALL name' noch einfach sein, doch dürfen bei 'ld hl,name' und 'ld hl,(name)' keine Zweifel entstehen.

Unterhalb der Bezeichnung einer Routine oder Speicherstelle sind jeweils die Referenzadressen angegeben. Man kann gleich erkennen, in welcher Form auf die Adresse zugegriffen wird: Steht hinter der Adresse ein '!', handelt es sich um einen CALL. Bei 1670› handelt es sich um einen lesenden Zugriff und bei 178A‹ um einen schreibenden. Steht hinter der Adresse ein ':' wird diese als Datum verwendet. Es hängt dann von der jeweiligen Routine ab, ob geschrieben, gelesen oder gesprungen wird. Steht ein Hochkomma' hinter der Adresse, handelt es sich um einen relativen Sprung (meist aus unmittelbarer Nähe). Um den Programmtext aber nicht unnötig zu zerreißen, wurden die meisten relativen Sprünge wieder aus dem Listing entfernt, nachdem die Zusammenhänge klar waren. Schließlich gibt es noch Adressen 'ohne was', hier handelt es sich um Sprünge (jp jp,m usw. oder rst).

Alle Zahlenangaben sind hexadezimal, Dezimalwerte haben einen Dezimalpunkt.

Die am Ende von Programmabschnitten enthaltenen Bytes C7 C7 C7 ... sind nicht verwendete ROM-Bereiche, ihr Inhalt ist ohne Bedeutung.

Jetzt muß ich noch auf ein besonderes Thema kommen: Die Ausführlichkeit der Kommentare. Natürlich kann man dort, wo man keine weiß, einfach keine hinschreiben. Es bleibt dann dem Leser überlassen, herauszufinden, ob an dieser Stelle 'alles selbstverständlich' ist oder ob dem Kommentator zu diesem Thema (noch) nichts eingefallen ist. Sie dürfen ruhig davon ausgehen, daß letzteres der Fall ist. Doch sollte man es auch zugeben und sich und anderen die Arbeit leichter machen. Deshalb stehen in diesem Listing an noch einigen Stellen immer drei Pünktchen '...', die fügt der Disassembler (absichtlich) immer dann ein, wenn er unter dieser Adresse nichts findet; will sagen: Hier könnte man noch etwas hinschreiben.

Doch nun stürzen Sie sich ins Listing. Fangen Sie am besten mit einem einfachen Thema, wie z.B. die Druckerausgabe an, damit Sie sich an die Syntax und den Stil gewöhnen. Es wird Ihnen später sicher gelingen, sich auch in komplexere Zusammenhänge einzuarbeiten. Ein umfangreicher Index und die Adress-Paginierung helfen Ihnen, die gewünschten Stellen schnell aufzufinden.

```
----- rst 0, SYSTEM RESET
      @ C00F
0000  01 89 7F     ld bc,7F89        Video Gate Array
0003  ED 49        out (c),c         select lower ROM
0005  C3 80 05     jp 0580           boot system

----- rst 1 <addr>, LOW JUMP, bit 14=lower, 15=upper ROM disabled
0008  C3 82 B9     jp B982           addr is in the range 0000 ... 3FFF

----- = jp(hl), low ROM or RAM, bit 14=lower, 15=upper ROM disabled
000B  C3 7C B9     jp B97C           addr is in the range 0000 ... 3FFF

----- jp(bc)
000E  C5           push bc
000F  C9            ret

----- rst 2, call to a sideways ROM <addr>, bit 14/15 select the ROM
0010  C3 16 BA     jp BA16

----- KL jp(hl) to a sideways ROM
0013  C3 10 BA     jp BA10

----- jp(de)
      @ 1443! 2CE3!
0016  D5           push de
0017  C9            ret

----- rst 3, FAR CALL (hl=param), <addr>,<ROM state>
0018  C3 BF B9     jp B9BF

----- jp(hl), FAR CALL, (hl)=addr, <c>=ROM select
001B  C3 B1 B9     jp B9B1

----- jp(hl)
001E  E9           jp (hl)

001F  00           nop

----- rst 4, RAM LAM, ld a,(hl) with ROMs disabled
0020  C3 CB BA     jp BACB

----- KL FAR ICALL, jp(hl=param), <addr><ROM state>
0023  C3 B9 B9     jp B9B9

0026  00 00

----- rst 5 <addr>, FIRM JUMP, jump to lower ROM
0028  C3 2E BA     jp BA2E

----- SAVE for CURRENT ROM STATE
002B  00

----- select ROM STATE
002C  ED 49        out (c),c
002E  D9           exx
002F  FB           ei

----- rst 6, USER RESTART
      @ 004E<
0030  F3           di
0031  D9           exx
0032  21 2B 00     ld hl,002B        SAVE for CURRENT ROM STATE
0035  71           ld (hl),c         save ROM STATE
0036  18 08        jr 0040
```

```
----- rst 7, INTERRUPT ENTRY
0038  C3 39 B9     jp B939          rst 7, INTERRUPT ENTRY

----- EXTERNAL INTERRUPT
      @ B977!
003B  C9           ret

003C  00           nop
003D  00           nop
003E  00           nop
003F  00           nop
0040  CB D1        set 2,c
0042  18 E8        jr 002C          select ROM STATE

----- copy 0000..0040 ROM to RAM, restore HI KERNEL JUMPBLOCK
      @ 0637!
0044  21 40 00     ld hl,0040       high ROM source
0047  2D           dec l
0048  7E           ld a,(hl)
0049  77           ld (hl),a
004A  20 FB        jr nz,0047       next
004C  3E C7        ld a,C7          =rst 0
004E  32 30 00     ld (0030),a      rst 6, USER RESTART
0051  21 91 03     ld hl,0391       data for HIGH KERNEL JUMPBLOCK (copied to BA
0054  11 00 B9     ld de,B900       destination
0057  01 E9 01     ld bc,01E9       number of bytes
005A  ED B0        ldir

----- KL CHOKE OFF, reset the kernel
      @ 05E9! BCC8!
005C  F3           di
005D  3A AB B1     ld a,(B1AB)      KL ROM state to call
0060  ED 5B A9 B1  ld de,(B1A9)     KL contains c006 = start of ROM
0064  06 C0        ld b,C0          count
0066  21 00 B1     ld hl,B100       start of firmware RAM
0069  36 00        ld (hl),00       set B100..B1C0 to 0
006B  23           inc hl
006C  10 FB        djnz 0069        next
006E  47           ld b,a
006F  0E FF        ld c,FF          =255.
0071  A9           xor c
0072  C0           ret nz
0073  4F           ld c,a           <c>=0
0074  5F           ld e,a           <e>=0
0075  57           ld d,a           <d>=0
0076  C9           ret

----- KL PREPARE TO CALL AN UPPER ROM; <c>=ROM sel, (hl)=entry addr 0=default
      @ 0659
0077  7C           ld a,h
0078  B5           or l             <hl>=0?
0079  79           ld a,c           ROM select
007A  20 04        jr nz,0080       there is another entry given
007C  7D           ld a,l
007D  21 06 C0     ld hl,C006       DEFAULT ENTRY TO UPPER ROM
0080  32 A8 B1     ld (B1A8),a      KL ROM select address
0083  32 AB B1     ld (B1AB),a      KL ROM state to call
0086  22 A9 B1     ld (B1A9),hl     KL contains c006 = start of ROM
0089  21 FF AB     ld hl,ABFF       default upper RAM boundary-1
008C  11 40 00     ld de,0040       lower RAM boundary
008F  01 FF B0     ld bc,B0FF       upper RAM boundary-1
0092  31 00 C0     ld sp,C000       init stack pointer
0095  DF A9 B1     rst 3,B1A9       KL contains c006 = start of ROM
0098  C7           rst 0            SYSTEM RESET on return
```

```
----- KL TIME PLEASE in <de,hl>
      @ BD0D!
0099  F3          di
009A  ED 5B 89 B1 ld de,(B189)      KL TIME byte 2,3
009E  2A 87 B1    ld hl,(B187)      KL TIME byte 0,1
00A1  FB          ei
00A2  C9          ret

----- KL TIME SET <de,hl>
      @ BD10!
00A3  F3          di
00A4  AF          xor a             reset time byte 4
00A5  32 8B B1    ld (B18B),a       KL TIME byte 4, (overflows after 116 years)
00A8  ED 53 89 B1 ld (B189),de      KL TIME byte 2,3
00AC  22 87 B1    ld (B187),hl      KL TIME byte 0,1
00AF  FB          ei
00B0  C9          ret

----- INTERRUPT SERVICE ROUTINE (every 1/300 second)
      @ B948!
00B1  21 87 B1    ld hl,B187        KL TIME byte 0,1
00B4  34          inc (hl)          sets z-flag if overflow occurs
00B5  23          inc hl            does not affect z-flag
00B6  28 FC       jr z,00B4         inc next higher clock byte
00B8  06 F5       ld b,F5           8522 port B
00BA  ED 78       in a,(c)          check for FRAME FLYBACK PULSE
00BC  1F          rra
00BD  30 08       jr nc,00C7        no FRAME FLYBACK pulse
00BF  2A 8C B1    ld hl,(B18C)      KL FRAME FLY LIST pointer
00C2  7C          ld a,h
00C3  B7          or a              is there a FRAME FLY ticker?
00C4  C4 53 01    call nz,0153      kick a ticker
00C7  2A 8E B1    ld hl,(B18E)      KL FAST TICKER LIST pointer
00CA  7C          ld a,h
00CB  B7          or a              is there a FAST TICKER?
00CC  C4 53 01    call nz,0153      kick a ticker
00CF  CD 61 1F    call 1F61         SOUND TICK (every 1/300 second)
00D2  21 92 B1    ld hl,B192        KL SLOW TICKER COUNT
00D5  35          dec (hl)
00D6  C0          ret nz            slow countdown not finished; return
00D7  36 06       ld (hl),06        300/6 = 1/50 second; recharge value
00D9  CD B7 1B    call 1BB7         KM update key state map (every 1/50 second)
00DC  2A 90 B1    ld hl,(B190)      KL pointer to TICK LIST
00DF  7C          ld a,h
00E0  B7          or a
00E1  C8          ret z             nothing on the TICK LIST
00E2  21 04 B1    ld hl,B104        KL INTERRUPT SERVICE CLASS
00E5  CB C6       set 0,(hl)        mark for a slow ticker
00E7  C9          ret

      @ 01F9
00E8  2B          dec hl
00E9  36 00       ld (hl),00        =0.
00EB  2B          dec hl
00EC  3A 01 B1    ld a,(B101)       KL INTERRUPT PENDING QUEUE (hibyte)
00EF  B7          or a
00F0  20 0C       jr nz,00FE        there is an event in the chain
00F2  22 00 B1    ld (B100),hl      KL INTERRUPT SERVICE QUEUE
00F5  22 02 B1    ld (B102),hl      KL INTERRUPT SERVICE CHAIN
00F8  21 04 B1    ld hl,B104        KL INTERRUPT SERVICE CLASS
00FB  CB F6       set 6,(hl)
00FD  C9          ret
```

```
00FE  ED 5B 02 B1 ld de,(B102)      KL INTERRUPT SERVICE CHAIN
0102  22 02 B1    ld (B102),hl      KL INTERRUPT SERVICE CHAIN
0105  EB          ex de,hl
0106  73          ld (hl),e
0107  23          inc hl
0108  72          ld (hl),d
0109  C9          ret

----- perform asyncronous event(s)
      @ B960!
010A  ED 73 05 B1 ld (B105),sp      KL save for SP on interrupt service
010E  31 87 B1    ld sp,B187        private interrupt stack
0111  E5          push hl
0112  D5           push de
0113  C5            push bc
0114  21 04 B1       ld hl,B104     KL INTERRUPT SERVICE CLASS
0117  CB 76          bit 6,(hl)     is it an Express Event?
0119  28 1E          jr z,0139      no, Normal Event
011B  CB FE          set 7,(hl)     mark as EXPRESS Event
011D  2A 00 B1       ld hl,(B100)   KL INTERRUPT SERVICE QUEUE
0120  7C             ld a,h
0121  B7             or a
0122  28 0E          jr z,0132      no more
0124  5E             ld e,(hl)
0125  23             inc hl
0126  56             ld d,(hl)
0127  ED 53 00 B1    ld (B100),de   KL INTERRUPT SERVICE QUEUE
012B  23             inc hl
012C  CD 0A 02       call 020A      ...
012F  F3             di
0130  18 EB          jr 011D        check for next in line

0132  21 04 B1       ld hl,B104     KL INTERRUPT SERVICE CLASS
0135  CB 46          bit 0,(hl)     test for a SLOW TICKER
0137  28 10          jr z,0149      no, none present
0139  36 00          ld (hl),00     =0.
013B  37             scf
013C  08             ex af,af'
013D  CD 89 01       call 0189      tick an event (called after: ex af,af')
0140  B7             or a
0141  08             ex af,af'
0142  21 04 B1       ld hl,B104     KL INTERRUPT SERVICE CLASS
0145  7E             ld a,(hl)
0146  B7             or a
0147  20 D2          jr nz,011B     there is another event outstanding
0149  36 00          ld (hl),00     =0.
014B  C1            pop bc
014C  D1           pop de
014D  E1          pop hl
014E  ED 7B 05 B1 ld sp,(B105)      KL save for SP on interrupt service
0152  C9          ret

----- kick a ticker
      @ 00C4! 00CC! 0161'
0153  5E          ld e,(hl)
0154  23          inc hl
0155  7E          ld a,(hl)
0156  23          inc hl
0157  B7          or a
0158  CA E2 01    jp z,01E2         KL EVENT, kick an event block (hl)
015B  57          ld d,a
015C  D5          push de
015D  CD E2 01     call 01E2        KL EVENT, kick an event block (hl)
0160  E1          pop hl
0161  18 F0       jr 0153           next
```

```
----- KL NEW FRAME FLY, (hl)=addr, <b>=class, <de,c>=far addr
      @ 0D4C  BCD7!
0163  E5          push hl
0164  23           inc hl
0165  23           inc hl
0166  CD D2 01     call 01D2         KL INIT EVENT BLOCK (hl)=block, <b>=class, <
0169  E1          pop hl

----- KL ADD FRAME FLY; (hl)=addr of block
      @ BCDA!
016A  11 8C B1    ld de,B18C         KL FRAME FLY LIST pointer
016D  C3 73 03    jp 0373            link a block (hl) onto list (de)

----- KL DEL FRAME FLY, remove a block (hl) from the list
      @ 0D40! 0D52! BCDD!
0170  11 8C B1    ld de,B18C         KL FRAME FLY LIST pointer
0173  C3 82 03    jp 0382            unlink a block (hl) from list (de)

----- KL NEW FAST TICKER, (hl)=block,<b>=class,<c>=ROM sel,(de)=event routine
      @ BCE0!
0176  E5          push hl
0177  23           inc hl
0178  23           inc hl
0179  CD D2 01     call 01D2         KL INIT EVENT BLOCK (hl)=block, <b>=class, <
017C  E1          pop hl

----- KL ADD FAST TICKER, put block (hl) onto list
      @ BCE3!
017D  11 8E B1    ld de,B18E         KL FAST TICKER LIST pointer
0180  C3 73 03    jp 0373            link a block (hl) onto list (de)

----- KL DEL FAST TICKER, remove block (hl) from the list
      @ BCE6!
0183  11 8E B1    ld de,B18E         KL FAST TICKER LIST pointer
0186  C3 82 03    jp 0382            unlink a block (hl) from list (de)

----- tick an event (called after: ex af,af')
      @ 013D!
0189  2A 90 B1    ld hl,(B190)       KL pointer to TICK LIST
018C  7C          ld a,h
018D  B7          or a
018E  C8          ret z              no more in chain
018F  5E          ld e,(hl)
0190  23          inc hl
0191  56          ld d,(hl)          de = tick chain
0192  23          inc hl
0193  4E          ld c,(hl)
0194  23          inc hl
0195  46          ld b,(hl)          bc = tick count
0196  78          ld a,b
0197  B1          or c
0198  28 16       jr z,01B0          count 0, block is dormant
019A  0B          dec bc             decrement tick count
019B  78          ld a,b
019C  B1          or c
019D  20 0E       jr nz,01AD         count not yet zero
019F  D5          push de
01A0  23           inc hl
01A1  23           inc hl
01A2  E5           push hl
01A3  23            inc hl
01A4  CD E2 01      call 01E2        KL EVENT, kick an event block (hl)
01A7  E1           pop hl
01A8  46           ld b,(hl)         get recharge count
01A9  2B           dec hl
```

```
01AA  4E           ld c,(hl)
01AB  2B           dec hl
01AC  D1          pop de
01AD  70          ld (hl),b        and store as tick count
01AE  2B          dec hl
01AF  71          ld (hl),c
01B0  EB          ex de,hl
01B1  18 D9       jr 018C          next on list

----- KL ADD TICKER, (hl)=tick block, <de>=initial count, <bc>=recharge value
      @ BCE9!
01B3  E5          push hl          (hl)=tick block
01B4  23           inc hl
01B5  23           inc hl
01B6  F3           di
01B7  73           ld (hl),e       <de>=initial count
01B8  23           inc hl
01B9  72           ld (hl),d
01BA  23           inc hl
01BB  71           ld (hl),c
01BC  23           inc hl
01BD  70           ld (hl),b       <bc>=recharge entry value
01BE  E1          pop hl
01BF  11 90 B1    ld de,B190       KL pointer to TICK LIST
01C2  C3 73 03    jp 0373          link a block (hl) onto list (de)

----- KL DEL TICKER, remove block (hl) from tick list
      @ BCEC!
01C5  11 90 B1    ld de,B190       KL pointer to TICK LIST
01C8  CD 82 03    call 0382        unlink a block (hl) from list (de)
01CB  D0          ret nc
01CC  EB          ex de,hl
01CD  23          inc hl
01CE  5E          ld e,(hl)
01CF  23          inc hl
01D0  56          ld d,(hl)
01D1  C9          ret

----- KL INIT EVENT BLOCK (hl)=block, <b>=class, <c>=ROM sel, (de)=routine
      @ 0166! 0179! 1C79! 1E78! BCEF!
01D2  F3          di
01D3  23          inc hl
01D4  23          inc hl
01D5  36 00       ld (hl),00
01D7  23          inc hl
01D8  70          ld (hl),b        <b>=event class
01D9  23          inc hl
01DA  73          ld (hl),e        <de>=address of routine
01DB  23          inc hl
01DC  72          ld (hl),d
01DD  23          inc hl
01DE  71          ld (hl),c        <c>=ROM selection
01DF  23          inc hl
01E0  FB          ei
01E1  C9          ret

----- KL EVENT, kick an event block (hl)
      @ 0158  015D! 01A4! 1C9B! 1F9C  20A5  2115  BCF2!
01E2  23          inc hl
01E3  23          inc hl
01E4  F3          di
01E5  7E          ld a,(hl)
01E6  34          inc (hl)
01E7  FA 06 02    jp m,0206        event disarmed, no action
01EA  B7          or a
```

```
01EB  20 13        jr nz,0200        other kicks are outstanding
01ED  23           inc hl
01EE  7E           ld a,(hl)
01EF  2B           dec hl
01F0  B7           or a
01F1  F2 2F 02     jp p,022F         KL link SYNC EVENT block (hl)=<addr>, <a>=cl
01F4  08           ex af,af'
01F5  30 12        jr nc,0209        ...
01F7  08           ex af,af'
01F8  87           add a,a
01F9  F2 E8 00     jp p,00E8         ...
01FC  23           inc hl
01FD  23           inc hl
01FE  18 23        jr 0223           ...

0200  08           ex af,af'
0201  38 01        jr c,0204         do not enable interrupts if carry
0203  FB           ei
0204  08           ex af,af'
0205  C9           ret

0206  35           dec (hl)
0207  18 F7        jr 0200           ...

0209  08           ex af,af'

      @ 012C!
020A  FB           ei
020B  7E           ld a,(hl)
020C  B7           or a
020D  F8           ret m
020E  E5           push hl
020F  CD 1C 02      call 021C        ...
0212  E1           pop hl
0213  35           dec (hl)
0214  C8           ret z
0215  F2 0E 02     jp p,020E         ...
0218  34           inc (hl)
0219  C9           ret

----- KL DO SYNC, perform SYNC EVENT block (hl)
      @ BCFE!
021A  23           inc hl            skip over chain and count
021B  23           inc hl

      @ 020F!
021C  23           inc hl
021D  7E           ld a,(hl)         get event class
021E  23           inc hl
021F  1F           rra               bit 0 set?
0220  D2 B9 B9     jp nc,B9B9        KL FAR ICALL, jp(hl=param), <addr><ROM state

      @ 01FE'
0223  5E           ld e,(hl)
0224  23           inc hl
0225  56           ld d,(hl)         (de)=routine address
0226  EB           ex de,hl
0227  E9           jp (hl)

----- KL SYNC RESET, clear synchronous event queue
      @ BCF5!
0228  21 00 00     ld hl,0000
022B  22 94 B1     ld (B194),hl      KL SYNC EVENT queue+1
022E  C9           ret
```

```
----- KL link SYNC EVENT block (hl)=<addr>, <a>=class
      @ 01F1  027F
022F  E5          push hl
0230  47           ld b,a
0231  11 96 B1     ld de,B196       KL sync event queue +3
0234  EB           ex de,hl
0235  2B           dec hl
0236  2B           dec hl
0237  56           ld d,(hl)
0238  2B           dec hl
0239  5E           ld e,(hl)        (de)=sync event queue
023A  7A           ld a,d
023B  B7           or a             end of queue?
023C  28 07        jr z,0245        yes, all done
023E  13           inc de
023F  13           inc de
0240  13           inc de
0241  1A           ld a,(de)        priority of new block
0242  B8           cp b
0243  30 EF        jr nc,0234       found block is of higher, try next
0245  D1          pop de            =new block
0246  1B          dec de
0247  23          inc hl
0248  7E          ld a,(hl)         set chain of old block
0249  12          ld (de),a          and put it in the new block
024A  1B          dec de
024B  72          ld (hl),d
024C  2B          dec hl
024D  7E          ld a,(hl)         put new block addr in old chain
024E  12          ld (de),a
024F  73          ld (hl),e
0250  08          ex af,af'
0251  38 01       jr c,0254         do not enable interrupts if carry
0253  FB          ei
0254  08          ex af,af'
0255  C9          ret

----- KL NEXT SYNC, =(hl), =<a> prev. prio, =carry
      @ BCFB!
0256  F3          di
0257  2A 93 B1    ld hl,(B193)      KL SYNC EVENT queue
025A  7C          ld a,h
025B  B7          or a
025C  28 17       jr z,0275         no SYNC EVENT on the queue
025E  E5          push hl
025F  5E           ld e,(hl)
0260  23           inc hl
0261  56           ld d,(hl)
0262  23           inc hl
0263  23           inc hl
0264  3A 95 B1     ld a,(B195)      KL EVENT CLASS
0267  BE           cp (hl)
0268  30 0A        jr nc,0274       skip
026A  F5           push af
026B  7E            ld a,(hl)
026C  32 95 B1      ld (B195),a     KL EVENT CLASS
026F  ED 53 93 B1   ld (B193),de    KL SYNC EVENT queue
0273  F1           pop af
0274  E1          pop hl
0275  FB          ei
0276  C9          ret
```

```
----- KL DONE SYNC, (hl)=block, <a>=prev. priority
      @ BD01!
0277  32 95 B1    ld (B195),a       KL EVENT CLASS
027A  23          inc hl
027B  23          inc hl
027C  35          dec (hl)          decrement count
027D  C8          ret z             return if zero
027E  F3          di
027F  F2 2F 02    jp p,022F         KL link SYNC EVENT block (hl)=<addr>, <a>=cl
0282  34          inc (hl)
0283  FB          ei
0284  C9          ret

----- KL DEL SYNC, delete block (hl) from queue
      @ 1C8A! BCF8!
0285  CD 8E 02    call 028E         KL DISARM EVENT block (hl)
0288  11 93 B1    ld de,B193        KL SYNC EVENT queue
028B  C3 82 03    jp 0382           unlink a block (hl) from list (de)

----- KL DISARM EVENT block (hl)
      @ 0285! BD0A!
028E  23          inc hl
028F  23          inc hl
0290  36 C0       ld (hl),C0        sets count negative
0292  2B          dec hl
0293  2B          dec hl
0294  C9          ret

----- KL EVENT DISABLE
      @ BD04!
0295  21 95 B1    ld hl,B195        KL EVENT CLASS
0298  CB EE       set 5,(hl)        set the disable bit
029A  C9          ret

----- KL EVENT ENABLE
      @ BD07!
029B  21 95 B1    ld hl,B195        KL EVENT CLASS
029E  CB AE       res 5,(hl)        reset the disable bit
02A0  C9          ret

----- KL LOG EXT, (bc)=RSX cmd table, (hl)=4 byte RAM area
      @ 035A! BCD1!
02A1  E5          push hl
02A2  ED 5B A6 B1 ld de,(B1A6)      KL RSX QUEUE
02A6  22 A6 B1    ld (B1A6),hl      KL RSX QUEUE
02A9  73          ld (hl),e
02AA  23          inc hl
02AB  72          ld (hl),d
02AC  23          inc hl
02AD  71          ld (hl),c
02AE  23          inc hl
02AF  70          ld (hl),b
02B0  E1          pop hl
02B1  C9          ret

----- KL FIND COMMAND (hl) in RSX or back ROM, =<c>ROM sel, =(hl)routine
      @ BCD4!
02B2  11 96 B1    ld de,B196        KL temp store for EXTERNAL COMMAND NAME on s
02B5  01 10 00    ld bc,0010        count
02B8  CD A6 BA    call BAA6         KL ldir, ROMs disabled
02BB  EB          ex de,hl
02BC  2B          dec hl
02BD  CB FE       set 7,(hl)
02BF  2A A6 B1    ld hl,(B1A6)      KL RSX QUEUE
02C2  7D          ld a,l
```

```
02C3  18 10       jr 02D5           ...

02C5  E5          push hl
02C6  23           inc hl
02C7  23           inc hl
02C8  4E           ld c,(hl)
02C9  23           inc hl
02CA  46           ld b,(hl)
02CB  CD F4 02     call 02F4        ...
02CE  D1          pop de
02CF  D8          ret c
02D0  EB          ex de,hl
02D1  7E          ld a,(hl)
02D2  23          inc hl
02D3  66          ld h,(hl)
02D4  6F          ld l,a
02D5  B4          or h
02D6  20 ED       jr nz,02C5        ...
02D8  0E FF       ld c,FF
02DA  0C          inc c
02DB  CD 83 BA    call BA83         KL ask CLASS <a> VERSION/MARK <hl> of ROM
02DE  F5          push af
02DF  E6 03        and 03
02E1  47           ld b,a
02E2  CC F4 02     call z,02F4      ...
02E5  38 09        jr c,02F0        ...
02E7  F1          pop af
02E8  87          add a,a
02E9  30 EF       jr nc,02DA        ...
02EB  79          ld a,c
02EC  B7          or a
02ED  28 EB       jr z,02DA         ...
02EF  C9          ret

02F0  F1          pop af
02F1  C3 0B 06    jp 060B           MC START FOREGROUND PROGRAM, (hl)=entry addr

      @ 02CB! 02E2!
02F4  21 04 C0    ld hl,C004        EXTERNAL COMMAND TABLE
02F7  78          ld a,b
02F8  B7          or a
02F9  28 04       jr z,02FF         table empty
02FB  60          ld h,b
02FC  69          ld l,c
02FD  0E FF       ld c,FF           disable upper ROM, disable lower ROM
02FF  CD 7E BA    call BA7E         KL SELECT an UPPER ROM <c>
0302  C5          push bc
0303  5E           ld e,(hl)
0304  23           inc hl
0305  56           ld d,(hl)
0306  23           inc hl
0307  EB           ex de,hl
0308  18 17        jr 0321          ...

030A  01 96 B1    ld bc,B196        KL temp store for EXTERNAL COMMAND NAME on s
030D  0A          ld a,(bc)
030E  BE          cp (hl)
030F  20 08       jr nz,0319        ...
0311  23          inc hl
0312  03          inc bc
0313  87          add a,a
0314  30 F7       jr nc,030D        ...
0316  EB          ex de,hl
0317  18 0C       jr 0325           ...
```

```
0319  7E          ld a,(hl)
031A  23          inc hl
031B  87          add a,a
031C  30 FB       jr nc,0319        ...
031E  13          inc de
031F  13          inc de
0320  13          inc de
0321  7E          ld a,(hl)
0322  B7          or a
0323  20 E5       jr nz,030A        ...
0325  C1          pop bc
0326  C3 8C BA    jp BA8C           KL restore previous ROM selection, <c>=prev.

----- KL ROM WALK, (de)=low, (hl)=hi avail. memory
      find and initialise BACKGROUND ROMs
      @ BCCB!
0329  0E 07       ld c,07           count of ROMs
032B  CD 32 03    call 0332         KL INIT BACKground ROM, <c>=ROM sel, <de>=lo
032E  0D          dec c
032F  20 FA       jr nz,032B        next ROM
0331  C9          ret

----- KL INIT BACKground ROM, <c>=ROM sel, <de>=lomem, <hl>=himem
      @ 032B! BCCE!
0332  79          ld a,c
0333  FE 08       cp 08             max # of BACKGROUND ROMs
0335  D0          ret nc
0336  CD 7E BA    call BA7E         KL SELECT an UPPER ROM <c>
0339  3A 00 C0    ld a,(C000)       ROM class
033C  E6 03       and 03
033E  3D          dec a
033F  20 1F       jr nz,0360        out of selectable range
0341  C5          push bc
0342  CD 06 C0     call C006        entry to upper ROM
0345  D5           push de
0346  23            inc hl
0347  EB            ex de,hl
0348  21 AA B1      ld hl,B1AA      B1AA=hibyte of ROM (C0), B1AA=ROM state
034B  ED 4B A8 B1   ld bc,(B1A8)    KL ROM select address
034F  06 00         ld b,00         clear hi part
0351  09            add hl,bc
0352  09            add hl,bc       <hl>=B1AA+ 2*<c>
0353  73            ld (hl),e
0354  23            inc hl
0355  72            ld (hl),d
0356  21 FC FF      ld hl,FFFC      4 bytes are reserved for LOG EXT
0359  19            add hl,de
035A  CD A1 02      call 02A1       KL LOG EXT, (bc)=RSX cmd table, (hl)=4 byte
035D  2B            dec hl
035E  D1           pop de
035F  C1          pop bc
0360  C3 8C BA    jp BA8C           KL restore previous ROM selection, <c>=prev.

----- find entry (de) within chain (hl)
      @ 0371' 0375! 0384!
0363  7E          ld a,(hl)         low byte from list
0364  BB          cp e              compare with block
0365  23          inc hl
0366  7E          ld a,(hl)         get hibyte
0367  2B          dec hl
0368  20 03       jr nz,036D        no match on lobyte, go
036A  BA          cp d              compare hibyte
036B  37          scf
036C  C8          ret z             if both match, return with carry
036D  B7          or a              is the hibyte zero?
```

```
036E  C8          ret z             yes, end of chain
036F  6E          ld l,(hl)         no, get new link into (hl)
0370  67          ld h,a
0371  18 F0       jr 0363           find entry (de) within chain (hl)

----- link a block (hl) onto list (de)
      @ 016D  0180  01C2
0373  EB          ex de,hl
0374  F3          di
0375  CD 63 03    call 0363         find entry (de) within chain (hl)
0378  38 06       jr c,0380         block already in the chain, return
037A  73          ld (hl),e
037B  23          inc hl
037C  72          ld (hl),d         insert the block address as the last link
037D  13          inc de
037E  AF          xor a
037F  12          ld (de),a         and mark the end of the chain
0380  FB          ei
0381  C9          ret                by setting the hibyte to zero

----- unlink a block (hl) from list (de)
      @ 0173  0186  01C8! 028B
0382  EB          ex de,hl
0383  F3          di
0384  CD 63 03    call 0363         find entry (de) within chain (hl)
0387  30 06       jr nc,038F        block not on the list, return
0389  1A          ld a,(de)         get the link entry
038A  77          ld (hl),a          and store it into the previous block
038B  13          inc de
038C  23          inc hl
038D  1A          ld a,(de)
038E  77          ld (hl),a
038F  FB          ei
0390  C9          ret

----- data for HIGH KERNEL JUMPBLOCK (copied to      ... BAE8)
      @ 0051:
0391  C3 5E BA C3 68 BA C3 4A  BA C3 54 BA C3 72 BA C3  7E BA C3 A2 BA C3 83 BA
03A9  C3 8C BA C3 A6 BA C3 AC  BA 3A 94 B1 B7 C8 E5 F3  2A 93 B1 7C B7 28 07 23
03C1  23 23 3A 95 B1 BE E1 FB  C9 F3 08 38 33 D9 79 37  FB 08 F3 F5 CB 91 ED 49
03D9  CD B1 00 B7 08 4F 06 7F  3A 04 B1 B7 28 14 FA 6A  B9 79 E6 0C F5 CB 91 D9
03F1  CD 0A 01 D9 E1 79 E6 F3  B4 4F ED 49 D9 F1 FB C9  08 E1 F5 CB D1 ED 49 CD
0409  3B 00 18 CF F3 E5 D9 D1  18 06 F3 D9 E1 5E 23 56  08 7A CB BA CB B2 07 07
0421  07 07 A9 E6 0C A9 C5 CD  A8 B9 F3 D9 08 79 C1 E6  03 CB 89 CB 81 B1 18 01
0439  D5 4F ED 49 B7 08 D9 FB  C9 F3 08 79 E5 D9 D1 18  15 F3 E5 D9 E1 18 09 F3
0451  D9 E1 5E 23 56 23 E5 EB  5E 23 56 23 08 7E FE FC  30 BE 06 DF ED 79 21 A8
0469  B1 46 77 C5 FD E5 3D FE  07 30 0F 87 C6 AC 6F CE  B1 95 67 7E 23 66 6F E5
0481  FD E1 06 7F 79 CB D7 CB  9F CD A8 B9 FD E1 F3 D9  08 59 C1 78 06 DF ED 79
0499  32 A8 B1 06 7F 7B 18 8F  F3 E5 D9 D1 18 08 F3 D9  E1 5E 23 56 23 E5 08 7A
04B1  CB FA CB F2 E6 C0 07 07  21 AB B1 86 18 A4 F3 D9  E1 5E 23 56 CB 91 ED 49
04C9  ED 53 3F BA D9 FB CD 3E  BA F3 D9 CB D1 ED 49 D9  FB C9 F3 D9 79 CB 91 ED
04E1  49 D9 FB C9 F3 D9 79 CB  D1 ED 49 D9 FB C9 F3 D9  79 CB 99 ED 49 D9 FB C9
04F9  F3 D9 79 CB D9 ED 49 D9  FB C9 F3 D9 A9 E6 0C A9  4F ED 49 D9 FB C9 CD 5E
0511  BA 18 0F CD 7E BA 3A 00  C0 2A 01 C0 F5 78 CD 72  BA F1 E5 F3 06 DF ED 49
0529  21 A8 B1 46 71 48 47 FB  E1 C9 3A A8 B1 C9 CD B2  BA ED B0 C9 CD B2 BA ED
0541  B8 C9 F3 D9 E1 C5 CB D1  CB D9 ED 49 CD C7 BA F3  D9 C1 ED 49 D9 FB C9 E5
0559  D9 FB C9 F3 D9 59 CB D3  CB DB ED 59 D9 7E D9 ED  49 D9 FB C9 D9 79 F6 0C
0571  ED 79 DD 7E 00 ED 49 D9  C9
_____________________________________________________________________
057A  C7 C7 C7 C7 C7 C7
```

```
----- boot system
      @ 0005
0580  F3          di
0581  01 82 F7    ld bc,F782        8522 control
0584  ED 49       out (c),c
0586  01 00 F4    ld bc,F400        8522 port A
0589  ED 49       out (c),c
058B  01 00 F6    ld bc,F600        8522 port C
058E  ED 49       out (c),c
0590  01 7F EF    ld bc,EF7F        printer latch
0593  ED 49       out (c),c
0595  06 F5       ld b,F5           8522 port B, ask soldered jumpers
0597  ED 78       in a,(c)
0599  E6 10       and 10            bit 4 set?
059B  21 C4 05    ld hl,05C4        initialisation data 50 Hz
059E  20 03       jr nz,05A3        if jumper not in
05A0  21 D4 05    ld hl,05D4        initialisation data 60 Hz
05A3  01 0F BC    ld bc,BC0F        CRTC address
05A6  ED 49       out (c),c
05A8  2B          dec hl
05A9  7E          ld a,(hl)
05AA  04          inc b
05AB  ED 79       out (c),a
05AD  05          dec b
05AE  0D          dec c
05AF  F2 A6 05    jp p,05A6         next
05B2  18 20       jr 05D4           continue

----- initialisation data 50 Hz
      @ (=05C4-0F) 059B:
05B4  3F 28 2E 8E 26 00 19 1E  00 07 00 00 30 00 C0 00

----- initialisation data 60 Hz
      @ (=05D4-0F) 05A0:
05C4  3F 28 2E 8E 1F 06 19 1B  00 07 00 00 30 00 C0 00

----- continue
05D4  11 5C 06    ld de,065C        find brandname and print
05D7  21 00 00    ld hl,0000
05DA  18 32       jr 060E

----- MC BOOT PROGRAM, load and run FOREGROUND
      @ BD13!
05DC  31 00 C0    ld sp,C000        init stack pointer
05DF  E5          push hl
05E0  CD 68 1E     call 1E68        SOUND RESET
05E3  F3           di
05E4  01 FF F8     ld bc,F8FF       expansion bus
05E7  ED 49        out (c),c        reset peripherals
05E9  CD 5C 00     call 005C        KL CHOKE OFF, reset the kernel
05EC  E1          pop hl
05ED  D5          push de
05EE  C5           push bc
05EF  E5            push hl
05F0  CD 1E 1A        call 1A1E     KM RESET key manager
05F3  CD 88 10        call 1088     TXT RESET text VDU
05F6  CD B1 0A        call 0AB1     SCR RESET screen pack
05F9  CD 5E BA        call BA5E     KL current upper ROM enable, <a>=prevoius RO
05FC  E1            pop hl
05FD  CD 75 07      call 0775       = jp(hl)
0600  C1           pop bc
0601  D1          pop de
0602  38 07       jr c,060B         MC START FOREGROUND PROGRAM, (hl)=entry addr
0604  EB          ex de,hl
0605  48          ld c,b
```

```
0606  11 E8 06     ld de,06E8       program load failed
0609  18 03        jr 060E

----- MC START FOREGROUND PROGRAM, (hl)=entry addr, <c>=ROM selection
      @ 02F1  0602' BD16!
060B  11 26 07     ld de,0726       = ret
060E  F3           di
060F  ED 56        im 1
0611  D9           exx
0612  01 00 DF     ld bc,DF00       expansion ROM select
0615  ED 49        out (c),c
0617  01 FF F8     ld bc,F8FF       expansion bus
061A  ED 49        out (c),c
061C  21 00 B1     ld hl,B100       start of firmware RAM
061F  11 01 B1     ld de,B101
0622  01 FF 07     ld bc,07FF       count
0625  36 00        ld (hl),00       clear B100-B8FF to 0
0627  ED B0        ldir
0629  01 89 7F     ld bc,7F89       Video gate array
062C  ED 49        out (c),c
062E  D9           exx
062F  AF           xor a
0630  08           ex af,af'
0631  31 00 C0     ld sp,C000       init stack pointer
0634  E5           push hl
0635  C5            push bc
0636  D5             push de
0637  CD 44 00         call 0044    copy 0000..0040 ROM to RAM, restore HI KERNE
063A  CD 88 08         call 0888    JUMP RESTORE standard jumpblock
063D  CD E0 19         call 19E0    KM INITIALISE key manager
0640  CD 68 1E         call 1E68    SOUND RESET
0643  CD A0 0A         call 0AA0    SCR INITIAJ.ISE screen pack
0646  CD 78 10         call 1078    TXT INITIALISE text VDU
0649  CD B0 15         call 15B0    GRA INITIALISE graphics VDU
064C  CD 70 23         call 2370    CAS INITIALISE cassette manager
064F  CD E6 07         call 07E6    MC RESET PRINTER indirection
0652  FB               ei
0653  E1              pop hl
0654  CD 75 07        call 0775     jp(hl); hl=065C
0657  C1            pop bc
0658  E1           pop hl
0659  C3 77 00     jp 0077          KL PREPARE TO CALL AN UPPER ROM; <c>=ROM sel

----- find brandname and print
      @ 05D4: = 0654!
065C  CD 12 07     call 0712        find brand name
065F  CD EB 06     call 06EB        print message in (HL)
0662  21 6D 06     ld hl,066D       ' 64K Microcomputer ...
0665  CD EB 06     call 06EB        print message in (HL)
0668  21 93 06     ld hl,0693       ' (c)1984 Amstrad Consumer ....
066B  18 7E        jr 06EB          print message in (HL)

----- ' 64K Microcomputer ...
066D   20 36 34 4B 20 4D 69 63  72 6F 63 6F 6D 70 75 74      ' 64K Microcomput
067D   65 72 20 20 28 76 31 29  0D 0A 0D 0A 00               'er  (v1).....
068A   43 6F 70 79 72 69 67 68  74                           'Copyright

----- ' (c)1984 Amstrad Consumer ....
0693   20 A4 31 39 38 34 20 41  6D 73 74 72 61 64 20 43      ' $1984 Amstrad C
06A3   6F 6E 73 75 6D 65 72 20  45 6C 65 63 74 72 6F 6E      'onsumer Electron
06B3   69 63 73 20 70 6C 63 0D  0A 20 20 20 20 20 20 20      'ics plc..
06C3   20 20 20 20 61 6E 64 20  4C 6F 63 6F 6D 6F 74 69      '     and Locomoti
06D3   76 65 20 53 6F 66 74 77  61 72 65 20 4C 74 64 2E      've Software Ltd.
06E3   0D 0A 0D 0A 00                                        '.....
```

```
----- program load failed
06E8  21 F4 06     ld hl,06F4         '*** PROGRAM LOAD FAILED ***

----- print message in (HL)
      @ 065F! 0665! 066B' 06F2'
06EB  7E           ld a,(hl)
06EC  23           inc hl
06ED  B7           or a
06EE  C8           ret z
06EF  CD 00 14     call 1400          TXT OUTPUT char or ctl code <a> to VDU
06F2  18 F7        jr 06EB            print message in (HL)

----- '*** PROGRAM LOAD FAILED ***
06F4  2A 2A 2A 20 50 52 4F 47  52 41 4D 20 4C 4F 41 44         '*** PROGRAM LOAD
0704  20 46 41 49 4C 45 44 20  2A 2A 2A 0D 0A 00               ' FAILED ***...

----- find brand name
      @ 065C!
0712  06 F5        ld b,F5            8522 port B, ask soldered jumpers
0714  ED 78        in a,(c)
0716  2F           cpl
0717  E6 0E        and 0E
0719  0F           rrca
071A  21 27 07     ld hl,0727         'brand names
071D  3C           inc a
071E  47           ld b,a
071F  7E           ld a,(hl)
0720  23           inc hl
0721  B7           or a
0722  20 FB        jr nz,071F         next char
0724  10 F9        djnz 071F          next entry
0726  C9           ret

----- 'brand names
0727  41 72 6E 6F 6C 64 00                                     'Arnold.
072E  0A 20 41 6D 73 74 72 61  64 00                           '. Amstrad.
0738  0A 20 4F 72 69 6F 6E 00                                  '. Orion.
0740  0A 20 53 63 68 6E 65 69  64 65 72 00                     '. Schneider.
074C  0A 20 41 77 61 00                                        '. Awa.
0752  0A 20 53 6F 6C 61 76 6F  78 00                           '. Solavox.
075C  0A 20 53 61 69 73 68 6F  00                              '. Saisho.
0765  0A 20 54 72 69 75 6D 70  68 00                           '. Triumph.
076F  0A 20 49 73 70 00                                        '. Isp.

----- = jp(hl)
0775  E9           jp (hl)

----- MC SET SCREEN MODE <a>
      @ 0B2B  BD1C!
0776  FE 03        cp 03              max 2
0778  D0           ret nc             wrong argument
0779  F3           di
077A  D9           exx
077B  CB 89        res 1,c
077D  CB 81        res 0,c            bits 0+1 of c' contain screen mode
077F  B1           or c
0780  4F           ld c,a             set new mode
0781  ED 49        out (c),c          b'=7F=Video Gate Array
0783  FB           ei
0784  D9           exx
0785  C9           ret
```

```
----- MC CLEAR INKS to one colour, (de)=ink vector
      @ 0AA3! 0D58  BD22!
0786  C5           push bc
0787  D5            push de
0788  01 10 7F       ld bc,7F10      8522 control
078B  CD AB 07       call 07AB       do the output to 8522
078E  0E 00          ld c,00
0790  CD AB 07       call 07AB       do the output to 8522
0793  1B             dec de
0794  20 FA          jr nz,0790      next
0796  D1            pop de
0797  C1           pop bc
0798  C9           ret

----- MC SET INKS, (de)=ink vector
      @ 0D68! 0D73! BD25!
0799  C5           push bc
079A  D5            push de
079B  01 10 7F       ld bc,7F10      8522 control
079E  CD AB 07       call 07AB       do the output to 8522
07A1  0E 00          ld c,00
07A3  CD AB 07       call 07AB       do the output to 8522
07A6  20 FB          jr nz,07A3      next
07A8  D1            pop de
07A9  C1           pop bc
07AA  C9           ret

----- do the output to 8522
      @ 078B! 0790! 079E! 07A3!
07AB  ED 49        out (c),c
07AD  1A           ld a,(de)
07AE  13           inc de
07AF  E6 1F        and 1F
07B1  F6 40        or 40
07B3  ED 79        out (c),a
07B5  0C           inc c
07B6  79           ld a,c
07B7  FE 10        cp 10
07B9  C9           ret

----- MC WAIT FLYBACK
      @ 0E05! 0E54! 0E8D! BD19!
07BA  F5           push af
07BB  C5            push bc
07BC  06 F5          ld b,F5         8522 port B
07BE  ED 78          in a,(c)
07C0  1F             rra
07C1  30 FB          jr nc,07BE      wait for bit 0
07C3  C1            pop bc
07C4  F1           pop af
07C5  C9           ret

----- MC set SCREEN OFFSET, <a>=base, <hl>=offset
      @ 0B4D  BD1F!
07C6  C5           push bc
07C7  0F            rrca
07C8  0F            rrca
07C9  E6 30         and 30           make sure it's a valid 16k area
07CB  4F            ld c,a
07CC  7C            ld a,h
07CD  1F            rra
07CE  E6 03         and 03           make sure it's legal
07D0  B1            or c
07D1  01 0C BC      ld bc,BC0C       CRTC address
07D4  ED 49         out (c),c
```

```
07D6  04          inc b
07D7  ED 79       out (c),a
07D9  05          dec b
07DA  0C          inc c
07DB  ED 49       out (c),c
07DD  04          inc b
07DE  7C          ld a,h
07DF  1F          rra
07E0  7D          ld a,l
07E1  1F          rra
07E2  ED 79       out (c),a
07E4  C1          pop bc
07E5  C9          ret

----- MC RESET PRINTER indirection
      @ 064F! BD28!
07E6  21 EC 07    ld hl,07EC       data for printer jumpblock
07E9  C3 8A 0A    jp 0A8A          copy (hl) bytes to address (hl+1),(hl+2)

----- data for printer jumpblock
07EC  03 F1 BD C3 F8 07

----- MC PRINT CHAR <a> to Centronics port
      @ BD2B!
07F2  C5          push bc
07F3  CD F1 BD    call BDF1        MC WAIT PRINTER, print char <a> or time out
07F6  C1          pop bc
07F7  C9          ret

----- MC WAIT PRINTER, print char <a> or time out
      @ BDF1
07F8  01 32 00    ld bc,0032       count for timeout
07FB  CD 1B 08    call 081B        MC BUSY PRINTER, if port is busy, =carry
07FE  30 07       jr nc,0807       MC SEND char <a> to PRINTER
0800  10 F9       djnz 07FB        try again
0802  0D          dec c
0803  20 F6       jr nz,07FB       try again
0805  B7          or a
0806  C9          ret

----- MC SEND char <a> to PRINTER
      @ 07FE' BD31!
0807  C5          push bc
0808  06 EF       ld b,EF          Centronics latch
080A  E6 7F       and 7F           mask out bit 7
080C  ED 79       out (c),a
080E  F6 80       or 80            set bit 7 (strobe)
0810  F3          di
0811  ED 79       out (c),a        send strobe and character
0813  E6 7F       and 7F           reset bit 7 (strobe)
0815  FB          ei
0816  ED 79       out (c),a        send character without strobe
0818  C1          pop bc
0819  37          scf
081A  C9          ret

----- MC BUSY PRINTER, if port is busy, =carry
      @ 07FB! BD2E!
081B  C5          push bc
081C  4F          ld c,a
081D  06 F5       ld b,F5          8522 port B
081F  ED 78       in a,(c)
0821  17          rla
0822  17          rla
0823  79          ld a,c
```

```
0824  C1          pop bc
0825  C9          ret

----- MC SOUND REGISTER, send <a>=reg#, <c>=data
      @ 1EDE! 218C! 2217! 221D! 2223! 227C  22A8  232C! 2335  BD34!
0826  F3          di
0827  06 F4       ld b,F4            8522 port A
0829  ED 79       out (c),a
082B  06 F6       ld b,F6            8522 port C
082D  ED 78       in a,(c)
082F  F6 C0       or C0              set bits 6+7
0831  ED 79       out (c),a
0833  E6 3F       and 3F             preserve lower bits
0835  ED 79       out (c),a
0837  06 F4       ld b,F4            8522 port A
0839  ED 49       out (c),c
083B  06 F6       ld b,F6            8522 port C
083D  4F          ld c,a
083E  F6 80       or 80              set bit 7
0840  ED 79       out (c),a          8522 port C
0842  ED 49       out (c),c
0844  FB          ei
0845  C9          ret

----- ask keys pressed and set map
      @ 1BBD!
0846  01 0E F4    ld bc,F40E         8522 port A
0849  ED 49       out (c),c
084B  06 F6       ld b,F6            8522 port C
084D  ED 78       in a,(c)
084F  E6 30       and 30             mask out bits
0851  4F          ld c,a
0852  F6 C0       or C0              set bit 6+7
0854  ED 79       out (c),a
0856  ED 49       out (c),c
0858  04          inc b
0859  3E 92       ld a,92            10010010; set port A to input
085B  ED 79       out (c),a          F7, 8522 control
085D  C5          push bc
085E  CB F1        set 6,c
0860  06 F6        ld b,F6           8522 port
0862  ED 49        out (c),c
0864  06 F4        ld b,F4           8522 port A
0866  ED 78        in a,(c)
0868  46           ld b,(hl)
0869  77           ld (hl),a
086A  A0           and b
086B  2F           cpl
086C  12           ld (de),a
086D  23           inc hl
086E  13           inc de
086F  0C           inc c
0870  79           ld a,c
0871  E6 0F        and 0F            mask out
0873  FE 0A        cp 0A             all keys done?
0875  20 E9        jr nz,0860        next
0877  C1          pop bc
0878  3E 82       ld a,82            10000010; reset port A to output
087A  ED 79       out (c),a          8522 control
087C  05          dec b
087D  ED 49       out (c),c          8522 port C
087F  C9          ret
```

```
0880  C7 C7 C7 C7 C7 C7 C7 C7

----- JUMP RESTORE standard jumpblock
      @ 063A! BD37!
0888  11 AC 08    ld de,08AC       data for STANDARD JUMPBLOCK (copied to BB00
088B  21 00 BB    ld hl,BB00       RAM address
088E  01 CF BF    ld bc,BFCF       <b>=count, CF = rst 1
0891  CD 97 08    call 0897        set up jump block
0894  01 EF 30    ld bc,30EF       <b>=count, EF = rst 5

----- set up jump block
      @ 0891! 08A9'
0897  71          ld (hl),c
0898  23          inc hl
0899  1A          ld a,(de)
089A  77          ld (hl),a
089B  13          inc de
089C  23          inc hl
089D  EB          ex de,hl
089E  79          ld a,c
089F  2F          cpl
08A0  07          rlca
08A1  07          rlca
08A2  E6 80       and 80           provide for ROM selection
08A4  B6          or (hl)
08A5  EB          ex de,hl
08A6  77          ld (hl),a
08A7  13          inc de
08A8  23          inc hl
08A9  10 EC       djnz 0897        next
08AB  C9          ret

----- data for STANDARD JUMPBLOCK (copied to BB00 ... )
      @ 0888:
08AC  E0 19 1E 1A 3C 1A 42 1A  77 1A BD 1A 2E 1B 7B 1A  56 1B 5C 1B BD 1C B3 1B
08C4  5C 1C 52 1D 3E 1D 57 1D  43 1D 5C 1D 48 1D AB 1C  A6 1C 6D 1C 69 1C 71 1C
08DC  82 1C 90 1C 78 10 88 10  51 14 4B 14 00 14 34 13  AB 13 A7 13 0C 12 56 12
08F4  40 15 5E 11 69 11 74 11  80 11 89 12 9A 12 79 12  81 12 CE 11 68 12 68 12
090C  A9 12 BD 12 AE 12 C3 12  C9 12 7A 13 87 13 D3 12  F1 12 FD 12 2A 13 CB 14
0924  E8 10 07 11 B0 15 DF 15  F4 15 F1 15 FC 15 04 16  12 16 34 17 79 17 A6 17
093C  BC 17 C5 17 F6 17 04 18  FD 17 0A 18 13 18 10 18  27 18 24 18 39 18 36 18
0954  45 19 A0 0A B1 0A 3C 0B  45 0B 50 0B CA 0A EC 0A  F7 0A 57 0B 64 0B A9 0B
096C  F9 0B 05 0C 13 0C 2D 0C  86 0C A0 0C EC 0C 14 0D  F1 0C 19 0D E4 0C E8 0C
0984  B3 0D B7 0D DF 0D FA 0D  3E 0E F3 0E 49 0F 49 0C  6B 0C C4 0F 2F 10 70 23
099C  7F 23 8E 23 4B 2A 4F 2A  51 2A 92 23 FC 23 01 24  35 24 AB 24 9A 24 96 24
09B4  AB 23 15 24 2E 24 5B 24  EA 24 28 25 3F 28 36 28  51 28 68 1E 9F 1F 6C 20
09CC  89 20 4A 20 CB 1E E6 1E  38 23 3D 23 49 23 4E 23  5C 00 29 03 32 03 A1 02
09E4  B2 02 63 01 6A 01 70 01  76 01 7D 01 83 01 B3 01  C5 01 D2 01 E2 01 28 02
09FC  85 02 56 02 1A 02 77 02  95 02 9B 02 8E 02 99 00  A3 00 DC 05 0B 06 BA 07
0A14  76 07 C6 07 86 07 99 07  E6 07 F2 07 1B 08 07 08  26 08 88 08 98 2A 18 2E
0A2C  29 2E 55 2E 66 2E 8E 2E  A1 2E AC 2E B6 2E 1D 2F  3F 33 37 33 3B 33 15 34
0A44  9E 34 78 35 9A 35 F8 35  E8 35 AE 31 A3 31 0A 31  0D 31 14 30 0F 30 90 30
0A5C  BC 31 B2 31 31 32 41 32  5E 2E 94 2F A1 2F B7 2F  E6 2F 08 37 0E 37 15 37
0A74  28 37 31 37 30 37 39 37  7A 37 81 37 50 37 8C 37  E9 37 D4 37 E0 37

----- copy (hl) bytes to address (hl+1),(hl+2)
      @ 07E9  0AB8! 108B! 15E2  1A30!
0A8A  4E          ld c,(hl)
0A8B  06 00       ld b,00          <line end>
0A8D  23          inc hl
0A8E  5E          ld e,(hl)
0A8F  23          inc hl
0A90  56          ld d,(hl)
0A91  23          inc hl
0A92  ED B0       ldir
```

```
0A94  C9          ret

0A95  C7 C7 C7 C7 C7 C7 C7 C7  C7 C7 C7

----- SCR INITIALISE screen pack
      @ 0643! BBFF!
0AA0  11 4D 10    ld de,104D        ink colours, flash period 1
0AA3  CD 86 07    call 0786         MC CLEAR INKS to one colour, (de)=ink vector
0AA6  3E C0       ld a,C0           default value for
0AA8  32 CB B1    ld (B1CB),a       SCR base of RAM for screen
0AAB  CD B1 0A    call 0AB1         SCR RESET screen pack
0AAE  C3 F2 0A    jp 0AF2           set up pixel bit map for mode 1

----- SCR RESET screen pack
      @ 05F6! 0AAB! BC02!
0AB1  AF          xor a             set to write mode 0 (FORCE mode)
0AB2  CD 49 0C    call 0C49         SCR ACCESS, set write mode <a> for graph VDU
0AB5  21 BE 0A    ld hl,0ABE        data for SCREEN PACK JUMPBLOCK
0AB8  CD 8A 0A    call 0A8A         copy (hl) bytes to address (hl+1),(hl+2)
0ABB  C3 D2 0C    jp 0CD2           reset to default inks and times of flash per

----- data for SCREEN PACK JUMPBLOCK
0ABE  09 E5 BD C3 82 0C C3 68  0C C3 F7 0A

----- SCR SET MODE <a>
      @ BC0E!
0ACA  E6 03       and 03            make sure it's legal
0ACC  FE 03       cp 03             max = 2
0ACE  D0          ret nc
0ACF  F5          push af
0AD0  CD 4F 0D     call 0D4F        set all inks to present flashing period
0AD3  F1          pop af
0AD4  5F          ld e,a
0AD5  CD B7 10    call 10B7         initialise all inks for 8 textstreams
0AD8  F5          push af
0AD9  CD D6 15     call 15D6        ...
0ADC  E5           push hl
0ADD  7B            ld a,e
0ADE  CD 11 0B      call 0B11       set up current pixel bitmap, mode=<a>
0AE1  CD EB BD      call BDEB       SCR CLEAR screen to ink 0
0AE4  E1           pop hl
0AE5  CD B6 15     call 15B6        ...
0AE8  F1          pop af
0AE9  C3 D5 10    jp 10D5           ...

----- SCR GET MODE <a>, cp 01
      @ 0B57! 0B65! 0BC3! 0CC3! 0EF3! 0F4D! 1621! 174D! 17AD! 1995! BC11!
0AEC  3A C8 B1    ld a,(B1C8)       SCR screen mode
0AEF  FE 01       cp 01
0AF1  C9          ret

----- set up pixel bit map for mode 1
      @ 0AAE
0AF2  3E 01       ld a,01           default mode 1
0AF4  CD 11 0B    call 0B11         set up current pixel bitmap, mode=<a>

----- SCR CLEAR screen to ink 0
      @ BC14! BDEB
0AF7  CD 4F 0D    call 0D4F         set all inks to present flashing period
0AFA  21 00 00    ld hl,0000
0AFD  CD 3C 0B    call 0B3C         SCR SET OFFSET (hl) of screen start
0B00  2A CA B1    ld hl,(B1CA)      SCR offset to screen start
0B03  2E 00       ld l,00           =0.
0B05  54          ld d,h
0B06  1E 01       ld e,01           =1.
```

```
0B08  01 FF 3F     ld bc,3FFF          count of screen bytes
0B0B  75           ld (hl),l
0B0C  ED B0        ldir
0B0E  C3 3C 0D     jp 0D3C             ...

----- set up current pixel bitmap, mode=<a>
      @ 0ADE! 0AF4!
0B11  21 3A 0B     ld hl,0B3A          mask for four pixels in a byte
0B14  FE 01        cp 01               is it mode 1?
0B16  38 08        jr c,0B20           no it's 0
0B18  21 36 0B     ld hl,0B36          mask for two pixels in a byte
0B1B  28 03        jr z,0B20           it's 1
0B1D  21 2E 0B     ld hl,0B2E          mask for one pixel in a byte
0B20  11 CF B1     ld de,B1CF          SCR current pixel bit map
0B23  01 08 00     ld bc,0008          byte count to copy
0B26  ED B0        ldir
0B28  32 C8 B1     ld (B1C8),a         SCR screen mode
0B2B  C3 76 07     jp 0776             MC SET SCREEN MODE <a>

----- mask for one pixel in a byte
0B2E  80 40 20 10 08 04 02 01

----- mask for two pixels in a byte
0B36  88 44 22 11

----- mask for four pixels in a byte
0B3A  AA 55

----- SCR SET OFFSET (hl) of screen start
      @ 0AFD! 0E3B  BC05!
0B3C  7C           ld a,h
0B3D  E6 07        and 07              make sure it's legal
0B3F  67           ld h,a
0B40  22 C9 B1     ld (B1C9),hl        SCR SCREEN START
0B43  18 05        jr 0B4A

----- SCR SET BASE of screen RAM <a>
      @ BC08!
0B45  E6 C0        and C0              make sure it's legal
0B47  32 CB B1     ld (B1CB),a         SCR base of RAM for screen
0B4A  CD 50 0B     call 0B50           SCR GET LOCATION of screen =<a>offset, =(hl)
0B4D  C3 C6 07     jp 07C6             MC set SCREEN OFFSET, <a>=base, <hl>=offset

----- SCR GET LOCATION of screen =<a>offset, =(hl)offset
      @ 0B4A! BC0B!
      @ 0B4A! BC0B!
0B50  2A C9 B1     ld hl,(B1C9)        SCR SCREEN START
0B53  3A CB B1     ld a,(B1CB)         SCR base of RAM for screen
0B56  C9           ret

----- SCR CHAR LIMITS, <b>=columns, <c>=lines
      @ 120C! BC17!
0B57  CD EC 0A     call 0AEC           SCR GET MODE <a>, cp 01
0B5A  01 18 13     ld bc,1318          19 columns, 24 lines
0B5D  D8           ret c
0B5E  06 27        ld b,27             39 columns
0B60  C8           ret z
0B61  06 4F        ld b,4F             79 columns
0B63  C9           ret

----- SCR CHAR POSITION conv phys coord to screen pos
      in: <hl>=col/row; out: (hl)=top left addr of char, <b>=width
      @ 0BA0! 0DE2! 0E49! 0E85! 0E89! 0F4A! 1357! BC1A!
0B64  D5           push de
0B65  CD EC 0A     call 0AEC           SCR GET MODE <a>, cp 01
```

```
0B68  06 04         ld b,04            4 pixels per byte
0B6A  38 05         jr c,0B71
0B6C  06 02         ld b,02            2 pixels per byte
0B6E  28 01         jr z,0B71
0B70  05            dec b              1 pixel per byte
0B71  C5            push bc
0B72  5C             ld e,h
0B73  16 00          ld d,00           <de>=<h>=column
0B75  62             ld h,d
0B76  D5             push de
0B77  54              ld d,h           de=hl=line
0B78  5D              ld e,l
0B79  29              add hl,hl
0B7A  29              add hl,hl
0B7B  19              add hl,de        4*line+line
0B7C  29              add hl,hl
0B7D  29              add hl,hl
0B7E  29              add hl,hl
0B7F  29              add hl,hl        (4*line+line)*16=80line
0B80  D1             pop de
0B81  19             add hl,de         +column * # of pixels = screen address
0B82  10 FD          djnz 0B81         add next
0B84  ED 5B C9 B1    ld de,(B1C9)      SCR SCREEN START
0B88  19             add hl,de
0B89  7C             ld a,h
0B8A  E6 07          and 07            make sure it's legal
0B8C  67             ld h,a
0B8D  3A CB B1       ld a,(B1CB)       SCR base of RAM for screen
0B90  84             add a,h
0B91  67             ld h,a
0B92  C1            pop bc
0B93  D1            pop de
0B94  C9            ret

      @ 0DB4! 0E44! 0E75!
0B95  7B            ld a,e
0B96  95            sub l
0B97  3C            inc a
0B98  87            add a,a
0B99  87            add a,a
0B9A  87            add a,a
0B9B  5F            ld e,a
0B9C  7A            ld a,d
0B9D  94            sub h
0B9E  3C            inc a
0B9F  57            ld d,a
0BA0  CD 64 0B      call 0B64          SCR CHAR POSITION conv phys coord to screen
0BA3  AF            xor a
0BA4  82            add a,d
0BA5  10 FD         djnz 0BA4          add next
0BA7  57            ld d,a
0BA8  C9            ret

----- SCR DOT POSITION convert base coordinates to screen position
      in: <de>=x, <hl>=y, out: (hl)=addr, <c>=pixel mask,  <b>=# of pixels
      @ 0FD3! 103B! 17E8! 181A! 1830! 196F! 19B6! BC1D!
0BA9  D5            push de
0BAA  EB             ex de,hl
0BAB  21 C7 00       ld hl,00C7        ...
0BAE  B7             or a
0BAF  ED 52          sbc hl,de
0BB1  7D             ld a,l
0BB2  E6 07          and 07            mask out
0BB4  87             add a,a
0BB5  87             add a,a
```

```
0BB6  87            add a,a
0BB7  4F            ld c,a
0BB8  7D            ld a,l
0BB9  E6 F8         and F8          11111000
0BBB  6F            ld l,a
0BBC  54            ld d,h
0BBD  5D            ld e,l
0BBE  29            add hl,hl
0BBF  29            add hl,hl
0BC0  19            add hl,de
0BC1  29            add hl,hl
0BC2  D1           pop de
0BC3  CD EC 0A     call 0AEC        SCR GET MODE <a>, cp 01
0BC6  06 01        ld b,01          =1.
0BC8  38 06        jr c,0BD0        ...
0BCA  06 03        ld b,03
0BCC  28 02        jr z,0BD0
0BCE  06 07        ld b,07          =7.
0BD0  78           ld a,b
0BD1  A3           and e
0BD2  F5           push af
0BD3  78            ld a,b
0BD4  0F            rrca
0BD5  CB 3A         srl d
0BD7  CB 1B         rr e
0BD9  0F            rrca
0BDA  38 F9         jr c,0BD5       ...
0BDC  19            add hl,de
0BDD  ED 5B C9 B1   ld de,(B1C9)    SCR SCREEN START
0BE1  19            add hl,de
0BE2  7C            ld a,h
0BE3  E6 07         and 07          =7.
0BE5  67            ld h,a
0BE6  3A CB B1      ld a,(B1CB)     SCR base of RAM for screen
0BE9  84            add a,h
0BEA  81            add a,c
0BEB  67            ld h,a
0BEC  F1           pop af
0BED  E5           push hl
0BEE  16 00         ld d,00         =0.
0BF0  5F            ld e,a
0BF1  21 CF B1      ld hl,B1CF      SCR current pixel bit map
0BF4  19            add hl,de
0BF5  4E            ld c,(hl)
0BF6  EB            ex de,hl
0BF7  E1           pop hl
0BF8  C9           ret

----- SCR NEXT BYTE, step screen addr (hl) right one byte
      in: (hl)=old screen addr, out: (hl)=new addr
      @ 0DC0! 0DEC! 0EDB! 0EDF! 0F78! 0FB3! 0FFA! 100B! 1365! 197C! BC20!
0BF9  2C           inc l
0BFA  C0           ret nz
0BFB  24           inc h
0BFC  7C           ld a,h
0BFD  E6 07        and 07           mask out
0BFF  C0           ret nz
0C00  7C           ld a,h
0C01  D6 08        sub 08           =8.
0C03  67           ld h,a
0C04  C9           ret
```

```
----- SCR PREV BYTE, step screen addr (hl) left one byte
      in: (hl)=old screen addr, out: (hl)=new addr
      @ BC23!
0C05  7D           ld a,l
0C06  2D           dec l
0C07  B7           or a
0C08  C0           ret nz
0C09  7C           ld a,h
0C0A  25           dec h
0C0B  E6 07        and 07            mask out
0C0D  C0           ret nz
0C0E  7C           ld a,h
0C0F  C6 08        add a,08          =8.
0C11  67           ld h,a
0C12  C9           ret

----- SCR NEXT LINE, step screen addr (hl) down one line
      in: (hl)=old screen addr, out: (hl)=new addr
      @ 0DD8! 0DF3! 0E5E! 0E63! 0F5B! 0F93! 0FBE! 136E! 1987! BC26!
0C13  7C           ld a,h
0C14  C6 08        add a,08          =8.
0C16  67           ld h,a
0C17  E6 38        and 38
0C19  C0           ret nz
0C1A  7C           ld a,h
0C1B  D6 40        sub 40            =64.
0C1D  67           ld h,a
0C1E  7D           ld a,l
0C1F  C6 50        add a,50          =80.
0C21  6F           ld l,a
0C22  D0           ret nc
0C23  24           inc h
0C24  7C           ld a,h
0C25  E6 07        and 07            mask
0C27  C0           ret nz
0C28  7C           ld a,h
0C29  D6 08        sub 08            =8.
0C2B  67           ld h,a
0C2C  C9           ret

----- SCR PREV LINE, step screen addr (hl) up one line
      in: (hl)=old screen addr, out: (hl)=new addr
      @ 0E90! 0E95! 1044! BC29!
0C2D  7C           ld a,h
0C2E  D6 08        sub 08            =8.
0C30  67           ld h,a
0C31  E6 38        and 38            mask
0C33  FE 38        cp 38             =56.
0C35  C0           ret nz
0C36  7C           ld a,h
0C37  C6 40        add a,40          =64.
0C39  67           ld h,a
0C3A  7D           ld a,l
0C3B  D6 50        sub 50            =80.
0C3D  6F           ld l,a
0C3E  D0           ret nc
0C3F  7C           ld a,h
0C40  25           dec h
0C41  E6 07        and 07            mask
0C43  C0           ret nz
0C44  7C           ld a,h
0C45  C6 08        add a,08          =8.
0C47  67           ld h,a
0C48  C9           ret
```

```
----- SCR ACCESS, set write mode <a> for graph VDU
      @ 0AB2! BC59!
0C49  E6 03       and 03          make sure it's legal
0C4B  21 6B 0C    ld hl,0C6B      SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl
0C4E  28 0F       jr z,0C5F       = OR-mode: new ink or old ink
0C50  FE 02       cp 02           check write mode
0C52  21 72 0C    ld hl,0C72      write mode 3:XOR-mode, NEW=ink OR old
0C55  38 08       jr c,0C5F       set up a jump instruction according to write
0C57  21 77 0C    ld hl,0C77      write mode 2: AND-mode, NEW=ink AND old
0C5A  28 03       jr z,0C5F       set up a jump instruction according to write
0C5C  21 7D 0C    ld hl,0C7D      write mode 1: OR-mode, NEW=ink XOR old

----- set up a jump instruction according to write mode
0C5F  3E C3       ld a,C3         = jp
0C61  32 CC B1    ld (B1CC),a     SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl
0C64  22 CD B1    ld (B1CD),hl    set jump address
0C67  C9          ret

----- SCR WRITE pixel(s) (hl)=addr, <c>=mask, using curr graph write mode
      @ BDE8
0C68  C3 CC B1    jp B1CC         SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl

----- SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl)=scr addr, <b>=ink, <c>=mask
      @ 0C4B: 13A4  BC5C!
0C6B  7E          ld a,(hl)
0C6C  A8          xor b
0C6D  B1          or c
0C6E  A9          xor c
0C6F  A8          xor b
0C70  77          ld (hl),a
0C71  C9          ret

----- write mode 3:XOR-mode, NEW=inkXOR old
0C72  78          ld a,b
0C73  A1          and c
0C74  AE          xor (hl)
0C75  77          ld (hl),a
0C76  C9          ret

----- write mode 2: AND-mode, NEW=ink AND old
0C77  79          ld a,c
0C78  2F          cpl
0C79  B0          or b
0C7A  A6          and (hl)
0C7B  77          ld (hl),a
0C7C  C9          ret

----- write mode 1: OR-mode, NEW=ink OR old
0C7D  78          ld a,b
0C7E  A1          and c
0C7F  B6          or (hl)
0C80  77          ld (hl),a
0C81  C9          ret

----- SCR READ a pixel from the screen, (hl)=addr, <c>=mask
      @ BDE5
0C82  7E          ld a,(hl)
0C83  C3 AC 0C    jp 0CAC         ...

----- SCR INK ENCODE, in: <a>=ink#, out: <a>=encoded ink
      @ 12B6! 17F6! 17FD! BC2C!
0C86  C5          push bc
0C87  D5           push de
0C88  CD C2 0C      call 0CC2     ...
0C8B  5F            ld e,a
```

```
0C8C  06 08         ld b,08          =8.
0C8E  3A CF B1      ld a,(B1CF)      SCR current pixel bit map
0C91  4F            ld c,a
0C92  CB 0B         rrc e
0C94  17            rla
0C95  CB 09         rrc c
0C97  38 02         jr c,0C9B        ...
0C99  CB 03         rlc e
0C9B  10 F5         djnz 0C92        ...
0C9D  D1            pop de
0C9E  C1            pop bc
0C9F  C9            ret

----- SCR INK DECODE, in: <a>=encoded ink; out: <a>=ink#
      @ 12C0  12C6  1807  180D  BC2F!
0CA0  C5            push bc
0CA1  47            ld b,a
0CA2  3A CF B1      ld a,(B1CF)      SCR current pixel bit map
0CA5  4F            ld c,a
0CA6  78            ld a,b
0CA7  CD AC 0C      call 0CAC        ...
0CAA  C1            pop bc
0CAB  C9            ret

      @ 0C83  0CA7!
0CAC  D5            push de
0CAD  11 08 00      ld de,0008
0CB0  0F            rrca
0CB1  CB 12         rl d
0CB3  CB 09         rrc c
0CB5  38 02         jr c,0CB9        ...
0CB7  CB 1A         rr d
0CB9  1D            dec e
0CBA  20 F4         jr nz,0CB0       ...
0CBC  7A            ld a,d
0CBD  CD C2 0C      call 0CC2        ...
0CC0  D1            pop de
0CC1  C9            ret

      @ 0C88! 0CBD!
0CC2  57            ld d,a
0CC3  CD EC 0A      call 0AEC        SCR GET MODE <a>, cp 01
0CC6  7A            ld a,d
0CC7  D0            ret nc
0CC8  0F            rrca
0CC9  0F            rrca
0CCA  CE 00         adc a,00         add carry
0CCC  0F            rrca
0CCD  9F            sbc a,a
0CCE  E6 06         and 06           mask
0CD0  AA            xor d
0CD1  C9            ret

----- reset to default inks and times of flash period 1
      @ 0ABB
0CD2  21 4D 10      ld hl,104D       ink colours, flash period 1
0CD5  11 D9 B1      ld de,B1D9       SCR table of colours, flash period 1
0CD8  01 22 00      ld bc,0022       len of block
0CDB  ED B0         ldir
0CDD  AF            xor a            set to flashing period 1
0CDE  32 FB B1      ld (B1FB),a      SCR flag which flash period is on (1 or 2)
0CE1  21 0A 0A      ld hl,0A0A       default flashing periods 10.,10.
```

```
----- SCR SET FLASHING PERIODS <h,l>
      @ BC3E!
0CE4  22 D7 B1     ld (B1D7),hl      SCR time for flashing period 1
0CE7  C9           ret

----- SCR GET FLASHING PERIODS <h,l>
      @ BC41!
0CE8  2A D7 B1     ld hl,(B1D7)      SCR time for flashing period 1
0CEB  C9           ret

----- SCR SET colour of INK, <a>=ink#, <b,c>=colours
      @ 14EE  BC32!
0CEC  E6 0F        and 0F            make sure it's legal
0CEE  3C           inc a
0CEF  18 01        jr 0CF2

----- SCR SET BORDER, <b,c>=colours
      @ 14F5  BC38!
0CF1  AF           xor a
0CF2  5F           ld e,a
0CF3  78           ld a,b
0CF4  CD 0A 0D     call 0D0A         get colour hardware# in <hl>
0CF7  46           ld b,(hl)
0CF8  79           ld a,c
0CF9  CD 0A 0D     call 0D0A         get colour hardware# in <hl>
0CFC  4E           ld c,(hl)
0CFD  7B           ld a,e
0CFE  CD 2F 0D     call 0D2F         get pointers (hl), (de), to colour <a>
0D01  71           ld (hl),c
0D02  EB           ex de,hl
0D03  70           ld (hl),b
0D04  3E FF        ld a,FF           set
0D06  32 FC B1     ld (B1FC),a       SCR flag
0D09  C9           ret

----- get colour hardware# in <hl>
      @ 0CF4! 0CF9!
0D0A  E6 1F        and 1F            mask out
0D0C  C6 93        add a,93
0D0E  6F           ld l,a
0D0F  CE 0D        adc a,0D          <hl>=<a>+0D93 (table of colours)
0D11  95           sub l
0D12  67           ld h,a
0D13  C9           ret

----- SCR GET colour(s) of INK, =<b,c>
      @ BC35!
0D14  E6 0F        and 0F            make sure it's legal
0D16  3C           inc a
0D17  18 01        jr 0D1A

----- SCR GET colour of BORDER
      @ BC3B!
0D19  AF           xor a
0D1A  CD 2F 0D     call 0D2F         get pointers (hl), (de), to colour <a>
0D1D  1A           ld a,(de)
0D1E  5E           ld e,(hl)
0D1F  CD 24 0D     call 0D24         look up hardware# for colour <a>
0D22  41           ld b,c
0D23  7B           ld a,e
```

```
----- look up hardware# for colour <a>
      @ OD1F!
OD24  0E 00       ld c,00          =0.
OD26  21 93 OD    ld hl,OD93       table of colour hardware numbers
OD29  BE          cp (hl)
OD2A  C8          ret z
OD2B  23          inc hl
OD2C  OC          inc c
OD2D  18 FA       jr OD29          next

----- get pointers (hl), (de), to colour <a>
      @ OCFE! OD1A!
OD2F  5F          ld e,a
OD30  16 00       ld d,00          clear hi part
OD32  21 EA B1    ld hl,B1EA       SCR table of colours, flash period 2
OD35  19          add hl,de
OD36  EB          ex de,hl
OD37  21 EF FF    ld hl,FFEF       = -17.
OD3A  19          add hl,de
OD3B  C9          ret

      @ OBOE
OD3C  21 FE B1    ld hl,B1FE       SCR FRAME FLY LIST
OD3F  E5          push hl
OD40  CD 70 01     call 0170       KL DEL FRAME FLY, remove a block (hl) from t
OD43  CD 6D OD     call OD6D       change flash period, change colours
OD46  11 5B OD     ld de,OD5B      decrement flash timer
OD49  06 81        ld b,81         =129.
OD4B  E1          pop hl
OD4C  C3 63 01    jp 0163          KL NEW FRAME FLY, (hl)=addr, <b>=class, <de,

----- set all inks to present flashing period
      @ OAD0! OAF7!
OD4F  21 FE B1    ld hl,B1FE       SCR FRAME FLY LIST
OD52  CD 70 01    call 0170        KL DEL FRAME FLY, remove a block (hl) from t
OD55  CD 81 OD    call OD81        get colour table (de) of flash period 1 or 2
OD58  C3 86 07    jp 0786          MC CLEAR INKS to one colour, (de)=ink vector

----- decrement flash timer
      @ OD46:
OD5B  21 FD B1    ld hl,B1FD       time count for current flash period
OD5E  35          dec (hl)         decrement timer
OD5F  28 OC       jr z,OD6D        change flash period, change colours
OD61  2B          dec hl
OD62  7E          ld a,(hl)        B1FC
OD63  B7          or a
OD64  C8          ret z
OD65  CD 81 OD    call OD81        get colour table (de) of flash period 1 or 2
OD68  CD 99 07    call 0799        MC SET INKS, (de)=ink vector
OD6B  18 OF       jr OD7C

----- change flash period, change colours
      @ OD43! OD5F'
OD6D  CD 81 OD    call OD81        get colour table (de) of flash period 1 or 2
OD70  32 FD B1    ld (B1FD),a      time count for current flash period
OD73  CD 99 07    call 0799        MC SET INKS, (de)=ink vector
OD76  21 FB B1    ld hl,B1FB       SCR flag which flash period is on (1 or 2)
OD79  7E          ld a,(hl)
OD7A  2F          cpl              flip flag
OD7B  77          ld (hl),a         and store back
OD7C  AF          xor a            reset
OD7D  32 FC B1    ld (B1FC),a      SCR flag
OD80  C9          ret
```

```
----- get colour table (de) of flash period 1 or 2; <a>=time setup
      @ 0D55! 0D65! 0D6D!
0D81  11 EA B1     ld de,B1EA        SCR table of colours, flash period 2
0D84  3A FB B1     ld a,(B1FB)       SCR flag which flash period is on (1 or 2)
0D87  B7           or a
0D88  3A D8 B1     ld a,(B1D8)       SCR time for flashing period 2
0D8B  C8           ret z
0D8C  11 D9 B1     ld de,B1D9        SCR table of colours, flash period 1
0D8F  3A D7 B1     ld a,(B1D7)       flash period for colour 1
0D92  C9           ret

----- table of colour hardware numbers
      @ 0D26:
0D93  14 04 15 1C 18 1D 0C 05  0D 16 06 17 1E 00 1F 0E
0DA3  07 0F 12 02 13 1A 19 1B  0A 03 0B 01 08 09 10 11

----- SCR FILL BOX, <a>=ink, <hl,de>=corners
      @ 1569! 1580! 159A! BC44!
0DB3  4F           ld c,a
0DB4  CD 95 0B     call 0B95         ...

----- SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=width/height
      @ 0DDC' 0E34  0E70  17F0! BC47!
0DB7  E5           push hl
0DB8  7A            ld a,d
0DB9  CD E8 0E      call 0EE8        ...
0DBC  30 09         jr nc,0DC7       ...
0DBE  42            ld b,d
0DBF  71            ld (hl),c
0DC0  CD F9 0B      call 0BF9        SCR NEXT BYTE, step screen addr (hl) right o
0DC3  10 FA         djnz 0DBF        next character position
0DC5  18 10         jr 0DD7          ...

0DC7  C5            push bc
0DC8  D5             push de
0DC9  71              ld (hl),c
0DCA  15              dec d
0DCB  28 08           jr z,0DD5      ...
0DCD  4A              ld c,d
0DCE  06 00           ld b,00        hi part =0
0DD0  54              ld d,h
0DD1  5D              ld e,l
0DD2  13              inc de
0DD3  ED B0           ldir
0DD5  D1             pop de
0DD6  C1            pop bc
0DD7  E1           pop hl
0DD8  CD 13 0C     call 0C13         SCR NEXT LINE, step screen addr (hl) down on
0DDB  1D           dec e
0DDC  20 D9        jr nz,0DB7        SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=
0DDE  C9           ret

----- SCR CHAR INVERT, (hl)=char pos, <b,c>=inks
      @ 1272! BC4A!
0DDF  78           ld a,b
0DE0  A9           xor c
0DE1  4F           ld c,a
0DE2  CD 64 0B     call 0B64         SCR CHAR POSITION conv phys coord to screen
0DE5  16 08        ld d,08           =8.
0DE7  E5           push hl
0DE8  C5            push bc
0DE9  7E             ld a,(hl)
0DEA  A9             xor c
0DEB  77             ld (hl),a
0DEC  CD F9 0B       call 0BF9       SCR NEXT BYTE, step screen addr (hl) right o
```

```
0DEF  10 F8      djnz 0DE9       next
0DF1  C1         pop bc
0DF2  E1         pop hl
0DF3  CD 13 0C   call 0C13       SCR NEXT LINE, step screen addr (hl) down on
0DF6  15         dec d
0DF7  20 EE      jr nz,0DE7      next
0DF9  C9         ret

----- SCR HARDWARE SCROLL, <a>=ink for new line, <b>=0=down, else up
      @ 11C9! BC4D!
0DFA  4F         ld c,a
0DFB  C5         push bc
0DFC  11 D0 FF   ld de,FFD0      =-48.
0DFF  06 30      ld b,30         =+48.
0E01  CD 24 0E   call 0E24       fill new line with ink <a>
0E04  C1         pop bc
0E05  CD BA 07   call 07BA       MC WAIT FLYBACK
0E08  78         ld a,b
0E09  B7         or a
0E0A  20 0D      jr nz,0E19      down
0E0C  11 B0 FF   ld de,FFB0      <e>=-80.
0E0F  CD 37 0E   call 0E37       inc offset of SCREEN START by <de>
0E12  11 00 00   ld de,0000
0E15  06 20      ld b,20         =32.
0E17  18 0B      jr 0E24         fill new line with ink <a>

----- down
0E19  11 50 00   ld de,0050      =+80.
0E1C  CD 37 0E   call 0E37       inc offset of SCREEN START by <de>
0E1F  11 B0 FF   ld de,FFB0      =-80.
0E22  06 20      ld b,20         =32.

----- fill new line with ink <a>
      @ 0E01! 0E17'
0E24  2A C9 B1   ld hl,(B1C9)    SCR SCREEN START
0E27  19         add hl,de
0E28  7C         ld a,h
0E29  E6 07      and 07          make sure it's legal
0E2B  67         ld h,a
0E2C  3A CB B1   ld a,(B1CB)     SCR base of RAM for screen
0E2F  84         add a,h
0E30  67         ld h,a
0E31  50         ld d,b
0E32  1E 08      ld e,08         height
0E34  C3 B7 0D   jp 0DB7         SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=

----- inc offset of SCREEN START by <de>
      @ 0E0F! 0E1C!
0E37  2A C9 B1   ld hl,(B1C9)    SCR SCREEN START
0E3A  19         add hl,de
0E3B  C3 3C 0B   jp 0B3C         SCR SET OFFSET (hl) of screen start

----- SCR WINDOW SCROLL up or down
      (hl)=left top, (de)=right bottom, <a>=new ink, <b>=0=down, else up
      @ 11C5! BC50!
0E3E  F5         push af
0E3F  78         ld a,b
0E40  B7         or a
0E41  28 30      jr z,0E73       up
0E43  E5         push hl
0E44  CD 95 0B   call 0B95       ...
0E47  E3         ex (sp),hl
0E48  2C         inc l
0E49  CD 64 0B   call 0B64       SCR CHAR POSITION conv phys coord to screen
0E4C  4A         ld c,d
```

```
0E4D  7B              ld a,e
0E4E  D6 08           sub 08          =8.
0E50  47              ld b,a
0E51  28 17           jr z,0E6A       ...
0E53  D1             pop de
0E54  CD BA 07       call 07BA        MC WAIT FLYBACK
0E57  C5             push bc
0E58  E5              push hl
0E59  D5               push de
0E5A  CD A4 0E          call 0EA4     ...
0E5D  E1               pop hl
0E5E  CD 13 0C         call 0C13      SCR NEXT LINE, step screen addr (hl) down on
0E61  EB               ex de,hl
0E62  E1              pop hl
0E63  CD 13 0C        call 0C13       SCR NEXT LINE, step screen addr (hl) down on
0E66  C1             pop bc
0E67  10 EE          djnz 0E57        next
0E69  D5             push de

      @ 0E51' 0E7F' 0EA2'
0E6A  E1             pop hl
0E6B  51             ld d,c
0E6C  1E 08          ld e,08          =8.
0E6E  F1            pop af
0E6F  4F            ld c,a
0E70  C3 B7 0D      jp 0DB7           SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=

----- up
0E73  E5             push hl
0E74  D5              push de
0E75  CD 95 0B         call 0B95      ...
0E78  4A               ld c,d
0E79  7B               ld a,e
0E7A  D6 08            sub 08         =8.
0E7C  47               ld b,a
0E7D  D1              pop de
0E7E  E3              ex (sp),hl
0E7F  28 E9           jr z,0E6A       ...
0E81  C5              push bc
0E82  6B               ld l,e
0E83  54               ld d,h
0E84  1C               inc e
0E85  CD 64 0B         call 0B64      SCR CHAR POSITION conv phys coord to screen
0E88  EB               ex de,hl
0E89  CD 64 0B         call 0B64      SCR CHAR POSITION conv phys coord to screen
0E8C  C1              pop bc
0E8D  CD BA 07        call 07BA       MC WAIT FLYBACK
0E90  CD 2D 0C        call 0C2D       SCR PREV LINE, step screen addr (hl) up one
0E93  E5              push hl
0E94  EB               ex de,hl
0E95  CD 2D 0C         call 0C2D      SCR PREV LINE, step screen addr (hl) up one
0E98  E5               push hl
0E99  C5                push bc
0E9A  CD A4 0E           call 0EA4    ...
0E9D  C1                pop bc
0E9E  D1               pop de
0E9F  E1              pop hl
0EA0  10 EE           djnz 0E90       ...
0EA2  18 C6           jr 0E6A         ...

      @ 0E5A! 0E9A!
0EA4  06 00         ld b,00           =0.
0EA6  CD E6 0E      call 0EE6         ...
0EA9  38 16         jr c,0EC1         ...
0EAB  CD E6 0E      call 0EE6         ...
```

```
OEAE  30 25      jr nc,OED5       ...
OEB0  C5         push bc
OEB1  AF          xor a
OEB2  95          sub l
OEB3  4F          ld c,a
OEB4  ED B0       ldir
OEB6  C1         pop bc
OEB7  2F         cpl
OEB8  3C         inc a
OEB9  81         add a,c
OEBA  4F         ld c,a
OEBB  7C         ld a,h
OEBC  D6 08      sub 08           =8.
OEBE  67         ld h,a
OEBF  18 14      jr OED5          ...

OEC1  CD E6 OE   call OEE6        ...
OEC4  38 12      jr c,OED8        ...
OEC6  C5         push bc
OEC7  AF          xor a
OEC8  93          sub e
OEC9  4F          ld c,a
OECA  ED B0       ldir
OECC  C1         pop bc
OECD  2F         cpl
OECE  3C         inc a
OECF  81         add a,c
OED0  4F         ld c,a
OED1  7A         ld a,d
OED2  D6 08      sub 08           =8.
OED4  57         ld d,a
OED5  ED B0      ldir
OED7  C9         ret

OED8  41         ld b,c
OED9  7E         ld a,(hl)
OEDA  12         ld (de),a
OEDB  CD F9 OB   call OBF9        SCR NEXT BYTE, step screen addr (hl) right o
OEDE  EB         ex de,hl
OEDF  CD F9 OB   call OBF9        SCR NEXT BYTE, step screen addr (hl) right o
OEE2  EB         ex de,hl
OEE3  10 F4      djnz OED9        ...
OEE5  C9         ret

      @ OEA6! OEAB! OEC1!
OEE6  79         ld a,c
OEE7  EB         ex de,hl

      @ ODB9!
OEE8  3D         dec a
OEE9  85         add a,l
OEEA  D0         ret nc
OEEB  7C         ld a,h
OEEC  E6 07      and 07           =7.
OEEE  EE 07      xor 07           =7.
OEF0  C0         ret nz
OEF1  37         scf
OEF2  C9         ret

----- SCR UNPACK, (hl)=matrix address, (de)=destination
      @ 1352! BC53!
OEF3  CD EC 0A   call 0AEC        SCR GET MODE <a>, cp 01
OEF6  06 08      ld b,08          byte count
OEF8  38 31      jr c,0F2B        mode 0
OEFA  28 06      jr z,0F02        mode 1
```

```
0EFC  01 08 00    ld bc,0008        mode 2
0EFF  ED B0       ldir
0F01  C9          ret

----- mode 1
0F02  4E          ld c,(hl)
0F03  23          inc hl
0F04  E5          push hl
0F05  C5           push bc
0F06  06 04          ld b,04         =4.
0F08  21 CF B1       ld hl,B1CF      SCR current pixel bit map
0F0B  AF             xor a
0F0C  CB 01          rlc c
0F0E  30 01          jr nc,0F11      ...
0F10  B6             or (hl)
0F11  23             inc hl
0F12  10 F8          djnz 0F0C       ...
0F14  12             ld (de),a
0F15  13             inc de
0F16  06 04          ld b,04         =4.
0F18  21 CF B1       ld hl,B1CF      SCR current pixel bit map
0F1B  AF             xor a
0F1C  CB 01          rlc c
0F1E  30 01          jr nc,0F21      ...
0F20  B6             or (hl)
0F21  23             inc hl
0F22  10 F8          djnz 0F1C       ...
0F24  12             ld (de),a
0F25  13             inc de
0F26  C1           pop bc
0F27  E1          pop hl
0F28  10 D8       djnz 0F02         mode 1
0F2A  C9          ret

----- mode 0
0F2B  4E          ld c,(hl)
0F2C  23          inc hl
0F2D  E5          push hl
0F2E  C5           push bc
0F2F  06 04          ld b,04         =4.
0F31  AF             xor a
0F32  21 CF B1       ld hl,B1CF      SCR current pixel bit map
0F35  CB 01          rlc c
0F37  30 01          jr nc,0F3A      ...
0F39  7E             ld a,(hl)
0F3A  23             inc hl
0F3B  CB 01          rlc c
0F3D  30 01          jr nc,0F40      ...
0F3F  B6             or (hl)
0F40  12             ld (de),a
0F41  13             inc de
0F42  10 ED          djnz 0F31       ...
0F44  C1           pop bc
0F45  E1          pop hl
0F46  10 E3       djnz 0F2B         mode 0
0F48  C9          ret

----- SCR REPACK char matrix, <a>=ink to match, (hl)=char pos, (de)=dest.
      @ 13C8! 13D7! BC56!
0F49  4F          ld c,a
0F4A  CD 64 0B    call 0B64         SCR CHAR POSITION conv phys coord to screen
0F4D  CD EC 0A    call 0AEC         SCR GET MODE <a>, cp 01
0F50  06 08       ld b,08           =8.
0F52  38 45       jr c,0F99         mode 0
0F54  28 0B       jr z,0F61         mode 1
```

```
0F56  7E          ld a,(hl)
0F57  A9          xor c
0F58  2F          cpl
0F59  12          ld (de),a
0F5A  13          inc de
0F5B  CD 13 0C    call 0C13         SCR NEXT LINE, step screen addr (hl) down on
0F5E  10 F6       djnz 0F56         next
0F60  C9          ret

----- mode 1
0F61  E5          push hl
0F62  D5           push de
0F63  E5            push hl
0F64  7E             ld a,(hl)
0F65  A9             xor c
0F66  21 CF B1       ld hl,B1CF     SCR current pixel bit map
0F69  16 04          ld d,04        =4.
0F6B  F5             push af
0F6C  A6              and (hl)
0F6D  20 01           jr nz,0F70
0F6F  37              scf
0F70  CB 13           rl e
0F72  23              inc hl
0F73  F1             pop af
0F74  15             dec d
0F75  20 F4          jr nz,0F6B     next
0F77  E1            pop hl
0F78  CD F9 0B      call 0BF9       SCR NEXT BYTE, step screen addr (hl) right o
0F7B  7E            ld a,(hl)
0F7C  A9            xor c
0F7D  21 CF B1      ld hl,B1CF      SCR current pixel bit map
0F80  16 04         ld d,04         =4.
0F82  F5            push af
0F83  A6             and (hl)
0F84  20 01          jr nz,0F87     ...
0F86  37             scf
0F87  CB 13          rl e
0F89  23             inc hl
0F8A  F1            pop af
0F8B  15            dec d
0F8C  20 F4         jr nz,0F82      ...
0F8E  E1           pop hl
0F8F  73           ld (hl),e
0F90  EB           ex de,hl
0F91  13           inc de
0F92  E1          pop hl
0F93  CD 13 0C    call 0C13         SCR NEXT LINE, step screen addr (hl) down on
0F96  10 C9       djnz 0F61         mode 1
0F98  C9          ret

----- mode 0
0F99  E5          push hl
0F9A  D5           push de
0F9B  16 04         ld d,04         =4.
0F9D  7E            ld a,(hl)
0F9E  E5            push hl
0F9F  A9             xor c
0FA0  F5             push af
0FA1  21 CF B1        ld hl,B1CF    SCR current pixel bit map
0FA4  A6              and (hl)
0FA5  20 01           jr nz,0FA8    ...
0FA7  37              scf
0FA8  CB 13           rl e
0FAA  F1             pop af
0FAB  23             inc hl
```

```
0FAC  A6            and (hl)
0FAD  20 01         jr nz,0FB0     ...
0FAF  37            scf
0FB0  CB 13         rl e
0FB2  E1            pop hl
0FB3  CD F9 0B      call 0BF9      SCR NEXT BYTE, step screen addr (hl) right o
0FB6  15            dec d
0FB7  20 E4         jr nz,0F9D     ...
0FB9  E1            pop hl
0FBA  73            ld (hl),e
0FBB  EB            ex de,hl
0FBC  13            inc de
0FBD  E1            pop hl
0FBE  CD 13 0C      call 0C13      SCR NEXT LINE, step screen addr (hl) down on
0FC1  10 D6         djnz 0F99      mode 0
0FC3  C9            ret

----- SCR HORIZONTAL line plot, <a>=ink, de=xbase, bc=xend, hl=ybase
      @ 190D! BC5F!
0FC4  F5            push af
0FC5  E5            push hl
0FC6  7A            ld a,d
0FC7  2F            cpl
0FC8  67            ld h,a
0FC9  7B            ld a,e
0FCA  2F            cpl
0FCB  6F            ld l,a
0FCC  23            inc hl
0FCD  09            add hl,bc
0FCE  23            inc hl
0FCF  E3            ex (sp),hl
0FD0  AF            xor a
0FD1  93            sub e
0FD2  F5            push af
0FD3  CD A9 0B      call 0BA9      SCR DOT POSITION convert base coordinates to
0FD6  E5            push hl
0FD7  78            ld a,b
0FD8  2F            cpl
0FD9  6F            ld l,a
0FDA  26 FF         ld h,FF        =255.
0FDC  22 07 B2      ld (B207),hl ...
0FDF  E1            pop hl
0FE0  F1            pop af
0FE1  A0            and b
0FE2  47            ld b,a
0FE3  28 45         jr z,102A      ...
0FE5  E3            ex (sp),hl
0FE6  18 03         jr 0FEB        ...

0FE8  1A            ld a,(de)
0FE9  B1            or c
0FEA  4F            ld c,a
0FEB  2B            dec hl
0FEC  7C            ld a,h
0FED  B5            or l
0FEE  28 34         jr z,1024      ...
0FF0  13            inc de
0FF1  10 F5         djnz 0FE8      ...
0FF3  EB            ex de,hl
0FF4  E1            pop hl
0FF5  F1            pop af
0FF6  47            ld b,a
0FF7  CD E8 BD      call BDE8      SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin
0FFA  CD F9 0B      call 0BF9      SCR NEXT BYTE, step screen addr (hl) right o
0FFD  E5            push hl
```

```
0FFE  2A 07 B2    ld hl,(B207)      ...
1001  19          add hl,de
1002  30 0C       jr nc,1010        ...
1004  EB          ex de,hl
1005  E1          pop hl
1006  0E FF       ld c,FF           =255.
1008  CD E8 BD    call BDE8         SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin
100B  CD F9 0B    call 0BF9         SCR NEXT BYTE, step screen addr (hl) right o
100E  18 ED       jr 0FFD           ...

1010  7B          ld a,e
1011  B7          or a
1012  28 0E       jr z,1022         ...
1014  AF          xor a
1015  21 CF B1    ld hl,B1CF        SCR current pixel bit map
1018  B6          or (hl)
1019  23          inc hl
101A  1D          dec e
101B  20 FB       jr nz,1018        ...
101D  4F          ld c,a
101E  E1          pop hl
101F  C3 E8 BD    jp BDE8           SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin

1022  E1          pop hl
1023  C9          ret

1024  E1          pop hl
1025  F1          pop af
1026  47          ld b,a
1027  C3 E8 BD    jp BDE8           SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin

102A  D1          pop de
102B  F1          pop af
102C  47          ld b,a
102D  18 CE       jr 0FFD           ...

----- SCR VERTICAL line plot, <a>=ink, de=xbase, bc=yend, hl=ybase
      @ 1932! BC62!
102F  F5          push af
1030  E5          push hl
1031  7C          ld a,h
1032  2F          cpl
1033  67          ld h,a
1034  7D          ld a,l
1035  2F          cpl
1036  6F          ld l,a
1037  23          inc hl
1038  09          add hl,bc
1039  23          inc hl
103A  E3          ex (sp),hl
103B  CD A9 0B    call 0BA9         SCR DOT POSITION convert base coordinates to
103E  D1          pop de
103F  F1          pop af
1040  47          ld b,a
1041  CD E8 BD    call BDE8         SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin
1044  CD 2D 0C    call 0C2D         SCR PREV LINE, step screen addr (hl) up one
1047  1B          dec de
1048  7A          ld a,d
1049  B3          or e
104A  20 F5       jr nz,1041        ...
104C  C9          ret               next
```

```
----- ink colours, flash period 1
      @ 0AA0: 0CD2:
104D  04 04 0A 13 0C 0B 14 15  0D 06 1E 1F 07 12 19 04  17

----- ink colours, flash period 2
105E  04 04 0A 13 0C 0B 14 15  0D 06 1E 1F 07 12 19 0A  07

-----
106F  C7 C7 C7 C7 C7 C7 C7 C7  C7

----- TXT INITIALISE text VDU
      @ 0646! BB4E!
1078  CD 88 10     call 1088        TXT RESET text VDU
107B  AF           xor a
107C  32 95 B2     ld (B295),a      TXT flag for user matrix table
107F  21 01 00     ld hl,0001       default inks PAPER 00, PEN 01
1082  CD 3D 11     call 113D        set PAPER/PEN to <hl>, set default window
1085  C3 A3 10     jp 10A3          initialise all windows

----- TXT RESET text VDU
      @ 05F3! 1078! BB51!
1088  21 91 10     ld hl,1091       data for VDU jumpblock
108B  CD 8A 0A     call 0A8A        copy (hl) bytes to address (hl+1),(hl+2)
108E  C3 5B 14     jp 145B          TXT reset to default control code table

----- data for VDU jumpblock
      @ 1088:
1091  0F CD BD C3 63 12 C3 63  12 C3 4A 13 C3 C0 13 C3  0C 14

----- initialise all windows
      @ 1085
10A3  3E 08        ld a,08          # of textstreams
10A5  11 0D B2     ld de,B20D       TXT table for text stream parameters (8 time
10A8  21 85 B2     ld hl,B285       TXT CURSOR column/row
10AB  01 0F 00     ld bc,000F       count
10AE  ED B0        ldir
10B0  3D           dec a
10B1  20 F5        jr nz,10A8       next stream
10B3  32 0C B2     ld (B20C),a      TXT current text stream selected
10B6  C9           ret              select text stream #0

----- initialise all inks for 8 textstreams
      @ 0AD5!
10B7  3A 0C B2     ld a,(B20C)      TXT current text stream selected
10BA  4F           ld c,a
10BB  06 08        ld b,08          count of textstreams
10BD  78           ld a,b
10BE  3D           dec a
10BF  CD E8 10     call 10E8        TXT STREAM <a> SELECT, <a>=old text stream
10C2  CD D0 BD     call BDD0        TXT DRAW/UNDRAW CURSOR, if enabled
10C5  CD C3 12     call 12C3        TXT GET PAPER ink =<a>
10C8  32 90 B2     ld (B290),a      TXT PAPER ink
10CB  CD BD 12     call 12BD        TXT GET PEN ink, =<a>
10CE  32 8F B2     ld (B28F),a      TXT PEN ink
10D1  10 EA        djnz 10BD        next
10D3  79           ld a,c
10D4  C9           ret

      @ 0AE9
10D5  4F           ld c,a
10D6  06 08        ld b,08          =8.
10D8  78           ld a,b
10D9  3D           dec a
10DA  CD E8 10     call 10E8        TXT STREAM <a> SELECT, <a>=old text stream
10DD  C5           push bc
```

```
10DE  2A 8F B2     ld hl,(B28F)      TXT PEN ink
10E1  CD 3D 11     call 113D         set PAPER/PEN to <hl>, set default window
10E4  C1           pop bc
10E5  10 F1        djnz 10D8         next text stream
10E7  79           ld a,c

----- TXT STREAM <a> SELECT, <a>=old text stream
      @ 10BF! 10DA! 110C! 1120' BBB4!
10E8  E6 07        and 07            make sure it's legal
10EA  21 0C B2     ld hl,B20C        TXT current text stream selected
10ED  BE           cp (hl)
10EE  C8           ret z             no change, return
10EF  C5           push bc
10F0  D5            push de
10F1  4E             ld c,(hl)
10F2  77             ld (hl),a
10F3  47             ld b,a
10F4  79             ld a,c
10F5  CD 2A 11       call 112A       get start of textstream parameters
10F8  CD 22 11       call 1122       copy 15. bytes (hl) to (de)
10FB  78             ld a,b
10FC  CD 2A 11       call 112A       get start of textstream parameters
10FF  EB             ex de,hl
1100  CD 22 11       call 1122       copy 15. bytes (hl) to (de)
1103  79             ld a,c
1104  D1            pop de
1105  C1           pop bc
1106  C9           ret

----- TXT SWAP STREAMS <b> with <c>
      @ BBB7!
1107  3A 0C B2     ld a,(B20C)       TXT current text stream selected
110A  F5           push af
110B  79            ld a,c
110C  CD E8 10      call 10E8        TXT STREAM <a> SELECT, <a>=old text stream
110F  78            ld a,b
1110  32 0C B2      ld (B20C),a      TXT current text stream selected
1113  CD 2A 11      call 112A        get start of textstream parameters
1116  D5            push de
1117  79             ld a,c
1118  CD 2A 11       call 112A       get start of textstream parameters
111B  E1            pop hl
111C  CD 22 11      call 1122        copy 15. bytes (hl) to (de)
111F  F1           pop af
1120  18 C6        jr 10E8           TXT STREAM <a> SELECT, <a>=old text stream

----- copy 15. bytes (hl) to (de)
      @ 10F8! 1100! 111C!
1122  C5           push bc
1123  01 0F 00      ld bc,000F       count = 15.
1126  ED B0         ldir
1128  C1           pop bc
1129  C9           ret

----- get start of textstream parameters
      @ 10F5! 10FC! 1113! 1118!
112A  E6 07        and 07            make sure it's legal
112C  5F           ld e,a
112D  87           add a,a
112E  87           add a,a
112F  87           add a,a
1130  87           add a,a
1131  93           sub e
1132  C6 0D        add a,0D          <de>=<a>*15+B20D
1134  5F           ld e,a
```

```
1135  CE B2       adc a,B2
1137  93          sub e
1138  57          ld d,a
1139  21 85 B2    ld hl,B285       TXT CURSOR column/row
113C  C9          ret

----- set PAPER/PEN to <hl>, set default window
      @ 1082! 10E1!
113D  EB          ex de,hl
113E  3E 03       ld a,03          =3.
1140  32 8D B2    ld (B28D),a      TXT cursor enable flag (user)
1143  7A          ld a,d
1144  CD AE 12    call 12AE        TXT SET PAPER ink <a>
1147  7B          ld a,e
1148  CD A9 12    call 12A9        TXT SET PEN ink <a>
114B  AF          xor a            off
114C  CD A7 13    call 13A7        TXT SET GRAPHIC char write, <a>=0=OFF, FF=ON
114F  CD 7A 13    call 137A        TXT SET BACKground being written <a>
1152  21 00 00    ld hl,0000
1155  11 7F 7F    ld de,7F7F       default window = 127. x 127.
1158  CD 0C 12    call 120C        TXT SET WINDOW <hl>=left top, <de>=right bot
115B  C3 51 14    jp 1451          TXT VDU ENABLE

----- TXT SET cursor to COLUMN <a>
      @ 2787! BB6F!
115E  3D          dec a
115F  21 89 B2    ld hl,B289       TXT column, window left upper corner
1162  86          add a,(hl)
1163  2A 85 B2    ld hl,(B285)     TXT CURSOR column/row
1166  67          ld h,a
1167  18 0E       jr 1177          undraw cursor, if enabled

----- TXT SET cursor to ROW <a>
      @ BB72!
1169  3D          dec a
116A  21 88 B2    ld hl,B288       TXT row; window left upper corner
116D  86          add a,(hl)
116E  2A 85 B2    ld hl,(B285)     TXT CURSOR column/row
1171  6F          ld l,a
1172  18 03       jr 1177          undraw cursor, if enabled

----- TXT SET CURSOR, <hl>=column/row
      @ 153D  2CE0! 2CE7  2D03! 2D0E! 2D7E! BB75!
1174  CD 8A 11    call 118A        HL = HL + cursor address

----- undraw cursor, if enabled
      @ 1167' 1172' 1241  152D
1177  CD D0 BD    call BDD0        TXT DRAW/UNDRAW CURSOR, if enabled

----- draw cursor, if enabled
      @ 1527
117A  22 85 B2    ld (B285),hl     TXT CURSOR column/row
117D  C3 CD BD    jp BDCD          TXT DRAW/UNDRAW CURSOR, if enabled

----- TXT GET CURSOR position (hl), roll count <a>
      @ 271F! 2792! 2B52! 2BF3! 2CB1! 2CDC! 2CEC! 2CFB! 2D56! 2D6A! 2D77!
      @ 2D8A! 2DAD! 2DC3! 2DD9! 2DE9! BB78!
1180  2A 85 B2    ld hl,(B285)     TXT CURSOR column/row
1183  CD 97 11    call 1197        HL = HL - cursor address
1186  3A 8C B2    ld a,(B28C)      TXT roll count
1189  C9          ret
```

```
----- HL = HL + cursor address
      @ 1174! 11CE!
118A  3A 88 B2    ld a,(B288)     TXT row; window left upper corner
118D  3D          dec a
118E  85          add a,l
118F  6F          ld l,a
1190  3A 89 B2    ld a,(B289)     TXT column, window left upper corner
1193  3D          dec a
1194  84          add a,h
1195  67          ld h,a
1196  C9          ret

----- HL = HL - cursor address
      @ 1183! 11D5!
1197  3A 88 B2    ld a,(B288)     TXT row; window left upper corner
119A  95          sub l
119B  2F          cpl
119C  3C          inc a
119D  3C          inc a
119E  6F          ld l,a
119F  3A 89 B2    ld a,(B289)     TXT column, window left upper corner
11A2  94          sub h
11A3  2F          cpl
11A4  3C          inc a
11A5  3C          inc a
11A6  67          ld h,a
11A7  C9          ret

----- remove cursor, validate, scroll up
      @ 133B! 151D! 1530! 154F! 1584! 158E!
11A8  CD D0 BD    call BDD0       TXT DRAW/UNDRAW CURSOR, if enabled

----- validate cursor, scroll window up
      @ 126B!
11AB  2A 85 B2    ld hl,(B285)    TXT CURSOR column/row
11AE  CD DA 11    call 11DA       check whether cursor is within window, force
11B1  22 85 B2    ld (B285),hl    TXT CURSOR column/row
11B4  D8          ret c
11B5  E5          push hl
11B6  21 8C B2     ld hl,B28C     TXT roll count
11B9  78           ld a,b
11BA  87           add a,a
11BB  3C           inc a
11BC  86           add a,(hl)
11BD  77           ld (hl),a
11BE  CD 56 12     call 1256      TXT GET WINDOW size, <hl>=left top, <de>=rig
11C1  3A 90 B2     ld a,(B290)    TXT PAPER ink
11C4  F5           push af
11C5  DC 3E 0E      call c,0E3E   SCR WINDOW SCROLL up or down
11C8  F1           pop af
11C9  D4 FA 0D     call nc,0DFA   SCR HARDWARE SCROLL, <a>=ink for new line, <
11CC  E1          pop hl
11CD  C9          ret

----- TXT VALIDATE cursor position <hl> column/row
      @ 2C8C! 2CBA! 2D15! 2D5C! 2D91! 2D9C! 2DB2! 2DDD! BB87!
11CE  CD 8A 11    call 118A       HL = HL + cursor address
11D1  CD DA 11    call 11DA       check whether cursor is within window, force
11D4  F5          push af
11D5  CD 97 11     call 1197      HL = HL - cursor address
11D8  F1          pop af
11D9  C9          ret
```

```
----- check whether cursor is within window, force it in
      @ 11AE! 11D1!
11DA  3A 8B B2     ld a,(B28B)       TXT column, window right bottom corner
11DD  BC           cp h
11DE  F2 E6 11     jp p,11E6         ...
11E1  3A 89 B2     ld a,(B289)       TXT column, window left upper corner
11E4  67           ld h,a
11E5  2C           inc l
11E6  3A 89 B2     ld a,(B289)       TXT column, window left upper corner
11E9  3D           dec a
11EA  BC           cp h
11EB  FA F3 11     jp m,11F3         ...
11EE  3A 8B B2     ld a,(B28B)       TXT column, window right bottom corner
11F1  67           ld h,a
11F2  2D           dec l
11F3  3A 88 B2     ld a,(B288)       TXT row; window left upper corner
11F6  3D           dec a
11F7  BD           cp l
11F8  F2 06 12     jp p,1206         ...
11FB  3A 8A B2     ld a,(B28A)       TXT row, window right bottom corner
11FE  BD           cp l
11FF  37           scf
1200  F0           ret p
1201  6F           ld l,a
1202  06 FF        ld b,FF           =255.
1204  B7           or a
1205  C9           ret

1206  3C           inc a
1207  6F           ld l,a
1208  06 00        ld b,00           =0.
120A  B7           or a
120B  C9           ret

----- TXT SET WINDOW <hl>=left top, <de>=right bottom corner
      @ 1158! 1501  BB66!
120C  CD 57 0B     call 0B57         SCR CHAR LIMITS, <b>=columns, <c>=lines
120F  7C           ld a,h
1210  CD 44 12     call 1244         ...
1213  67           ld h,a
1214  7A           ld a,d
1215  CD 44 12     call 1244         ...
1218  57           ld d,a
1219  BC           cp h
121A  30 02        jr nc,121E        ...
121C  54           ld d,h
121D  67           ld h,a
121E  7D           ld a,l
121F  CD 4D 12     call 124D         ...
1222  6F           ld l,a
1223  7B           ld a,e
1224  CD 4D 12     call 124D         ...
1227  5F           ld e,a
1228  BD           cp l
1229  30 02        jr nc,122D        ...
122B  5D           ld e,l
122C  6F           ld l,a
122D  22 88 B2     ld (B288),hl      TXT row; window left upper corner
1230  ED 53 8A B2  ld (B28A),de      TXT row, window right bottom corner
1234  7C           ld a,h
1235  B5           or l
1236  20 06        jr nz,123E        ...
1238  7A           ld a,d
1239  A8           xor b
123A  20 02        jr nz,123E        ...
```

```
123C  7B          ld a,e
123D  A9          xor c
123E  32 87 B2    ld (B287),a       TXT window flag; 0=whole screen
1241  C3 77 11    jp 1177           undraw cursor, if enabled

      @ 1210! 1215!
1244  B7          or a
1245  F2 49 12    jp p,1249         ...
1248  AF          xor a
1249  B8          cp b
124A  D8          ret c
124B  78          ld a,b
124C  C9          ret

      @ 121F! 1224!
124D  B7          or a
124E  F2 52 12    jp p,1252         ...
1251  AF          xor a
1252  B9          cp c
1253  D8          ret c
1254  79          ld a,c
1255  C9          ret

----- TXT GET WINDOW size, <hl>=left top, <de>=right bottom corner
      @ 11BE! 278E! 2BEC! BB69!
1256  2A 88 B2    ld hl,(B288)      TXT row; window left upper corner
1259  ED 5B 8A B2 ld de,(B28A)      TXT row, window right bottom corner
125D  3A 87 B2    ld a,(B287)       TXT window flag; 0=whole screen
1260  C6 FF       add a,FF          set carry if window is not the whole screen
1262  C9          ret

----- TXT DRAW/UNDRAW CURSOR, if enabled
      @ 1097  BDCD  BDD0
1263  3A 8D B2    ld a,(B28D)       TXT cursor enable flag (user)
1266  B7          or a
1267  C0          ret nz

----- TXT PLACE/REMOVE CURSOR on screen
      @ 2CCD: 2CD2: 2D06! BB8A! BB8D!
1268  C5          push bc
1269  D5           push de
126A  E5            push hl
126B  CD AB 11       call 11AB      validate cursor, scroll window up
126E  ED 4B 8F B2    ld bc,(B28F)   TXT PEN ink
1272  CD DF 0D       call 0DDF      SCR CHAR INVERT, (hl)=char pos, <b,c>=inks
1275  E1            pop hl
1276  D1           pop de
1277  C1          pop bc
1278  C9          ret

----- TXT CURSOR ON
      @ 276E! 2DE6! BB81!
1279  F5          push af
127A  3E FD        ld a,FD          11111101
127C  CD 8B 12     call 128B        cursor enable (user)
127F  F1          pop af
1280  C9          ret

----- TXT CURSOR OFF
      @ 2774! 2DF3  BB84!
1281  F5          push af
1282  3E 02        ld a,02          00000010
1284  CD 9C 12     call 129C        cursor disable (user)
1287  F1          pop af
1288  C9          ret
```

```
----- TXT CURSOR ENABLE (user)
      @ 1451! BB7B!
1289  3E FE        ld a,FE             11111110

----- cursor enable (user)
      @ 127C!
128B  F5           push af
128C  CD D0 BD      call BDD0          TXT DRAW/UNDRAW CURSOR, if enabled
128F  F1           pop af
1290  E5           push hl
1291  21 8D B2      ld hl,B28D         TXT cursor enable flag (user)
1294  A6            and (hl)
1295  77            ld (hl),a
1296  E1           pop hl
1297  C3 CD BD     jp BDCD             TXT DRAW/UNDRAW CURSOR, if enabled

----- TXT CURSOR DISABLE (user)
      @ 144B! BB7E!
129A  3E 01        ld a,01             =1.

----- cursor disable (user)
      @ 1284!
129C  F5           push af
129D  CD D0 BD      call BDD0          TXT DRAW/UNDRAW CURSOR, if enabled
12A0  F1           pop af
12A1  E5           push hl
12A2  21 8D B2      ld hl,B28D         TXT cursor enable flag (user)
12A5  B6            or (hl)
12A6  77            ld (hl),a
12A7  E1           pop hl
12A8  C9           ret

----- TXT SET PEN ink <a>
      @ 1148! BB90!
12A9  21 8F B2     ld hl,B28F          TXT PEN ink
12AC  18 03        jr 12B1

----- TXT SET PAPER ink <a>
      @ 1144! BB96!
12AE  21 90 B2     ld hl,B290          TXT PAPER ink
12B1  F5           push af
12B2  CD D0 BD      call BDD0          TXT DRAW/UNDRAW CURSOR, if enabled
12B5  F1           pop af
12B6  CD 86 0C     call 0C86           SCR INK ENCODE, in: <a>=ink#, out: <a>=encod
12B9  77           ld (hl),a
12BA  C3 CD BD     jp BDCD             TXT DRAW/UNDRAW CURSOR, if enabled

----- TXT GET PEN ink, =<a>
      @ 10CB! BB93!
12BD  3A 8F B2     ld a,(B28F)         TXT PEN ink
12C0  C3 A0 0C     jp 0CA0             SCR INK DECODE, in: <a>=encoded ink; out: <a

----- TXT GET PAPER ink =<a>
      @ BB99!
12C3  3A 90 B2     ld a,(B290)         TXT PAPER ink
12C6  C3 A0 0C     jp 0CA0             SCR INK DECODE, in: <a>=encoded ink; out: <a

----- TXT INVERSE, swap PEN/PAPER ink
      @ BB9C!
12C9  2A 8F B2     ld hl,(B28F)        TXT PEN ink
12CC  7C           ld a,h
12CD  65           ld h,l
12CE  6F           ld l,a
12CF  22 8F B2     ld (B28F),hl        TXT PEN ink
12D2  C9           ret
```

```
----- TXT GET char <a> MATRIX, (hl)=address, carry=user
      @ 12F2! 1309! 134B! 13E6! 1947! BBA5!
12D3  D5            push de
12D4  5F             ld e,a
12D5  CD 2A 13       call 132A         TXT GET user MATRIX TABLE (hl)=addr, <a>=fir
12D8  30 09          jr nc,12E3        use standard symbols
12DA  57             ld d,a
12DB  7B             ld a,e
12DC  92             sub d
12DD  3F             ccf
12DE  30 03          jr nc,12E3        use standard symbols
12E0  5F             ld e,a
12E1  18 03          jr 12E6

----- use standard symbols
12E3  21 00 38       ld hl,3800        SYMBOL images, start of table
12E6  F5             push af
12E7  16 00           ld d,00          =0.
12E9  EB              ex de,hl
12EA  29              add hl,hl
12EB  29              add hl,hl
12EC  29              add hl,hl
12ED  19              add hl,de
12EE  F1             pop af
12EF  D1            pop de
12F0  C9            ret

----- TXT SET char MATRIX, <a>=char, (hl)=matrix to set
      @ 1507  BBA8!
12F1  EB            ex de,hl
12F2  CD D3 12      call 12D3          TXT GET char <a> MATRIX, (hl)=address, carry
12F5  D0            ret nc
12F6  EB            ex de,hl

----- copy 8 bytes (hl) to (de)
      @ 1315!
12F7  01 08 00      ld bc,0008
12FA  ED B0         ldir
12FC  C9            ret

----- TXT SET user MATRIX TABLE addr (de), (hl)=new table
      @ BBAB!
12FD  E5            push hl
12FE  7A             ld a,d
12FF  B7             or a
1300  16 00          ld d,00           =0.
1302  20 19          jr nz,131D        ...
1304  15             dec d
1305  D5             push de
1306  4B              ld c,e
1307  EB              ex de,hl
1308  79              ld a,c
1309  CD D3 12        call 12D3        TXT GET char <a> MATRIX, (hl)=address, carry
130C  7C              ld a,h
130D  AA              xor d
130E  20 04           jr nz,1314       ...
1310  7D              ld a,l
1311  AB              xor e
1312  28 08           jr z,131C        ...
1314  C5              push bc
1315  CD F7 12         call 12F7       copy 8 bytes (hl) to (de)
1318  C1              pop bc
1319  0C              inc c
131A  20 EC           jr nz,1308       ...
131C  D1             pop de
```

```
131D  CD 2A 13     call 132A       TXT GET user MATRIX TABLE (hl)=addr, <a>=fir
1320  ED 53 94 B2  ld (B294),de    TXT first char of user matrix table
1324  D1          pop de
1325  ED 53 96 B2 ld (B296),de     TXT start of user matrix table
1329  C9          ret

----- TXT GET user MATRIX TABLE (hl)=addr, <a>=first char in table
      @ 12D5! 131D! BBAE!
132A  2A 94 B2    ld hl,(B294)     TXT first char of user matrix table
132D  7C          ld a,h
132E  0F          rrca
132F  7D          ld a,l
1330  2A 96 B2    ld hl,(B296)     TXT start of user matrix table
1333  C9          ret

----- TXT WRITE char <a> to screen
      @ 1424  26D4! 2DBF! BB5D!
1334  47          ld b,a
1335  3A 8E B2    ld a,(B28E)      TXT flag VDU enable
1338  B7          or a
1339  C8          ret z
133A  C5          push bc
133B  CD A8 11     call 11A8       remove cursor, validate, scroll up
133E  24           inc h
133F  22 85 B2     ld (B285),hl    TXT CURSOR column/row
1342  25           dec h
1343  F1          pop af
1344  CD D3 BD    call BDD3        TXT WRITE CHAR <a> on screen, <hl>=pos
1347  C3 CD BD    jp BDCD          TXT DRAW/UNDRAW CURSOR, if enabled

----- TXT WRITE CHAR <a> on screen, <hl>=pos
      @ BDD3
134A  E5          push hl
134B  CD D3 12     call 12D3       TXT GET char <a> MATRIX, (hl)=address, carry
134E  11 98 B2     ld de,B298      TXT buffer for unpacked char matrix
1351  D5           push de
1352  CD F3 0E      call 0EF3      SCR UNPACK, (hl)=matrix address, (de)=destin
1355  D1           pop de
1356  E1          pop hl
1357  CD 64 0B    call 0B64        SCR CHAR POSITION conv phys coord to screen
135A  0E 08       ld c,08
135C  C5          push bc
135D  E5           push hl
135E  C5            push bc
135F  D5             push de
1360  EB               ex de,hl
1361  4E               ld c,(hl)
1362  CD 76 13         call 1376   write BACKGROUND/FOREGROUND
1365  CD F9 0B         call 0BF9   SCR NEXT BYTE, step screen addr (hl) right o
1368  D1             pop de
1369  13             inc de
136A  C1            pop bc
136B  10 F1         djnz 135E      ...
136D  E1           pop hl
136E  CD 13 0C     call 0C13       SCR NEXT LINE, step screen addr (hl) down on
1371  C1          pop bc
1372  0D          dec c
1373  20 E7       jr nz,135C       ...
1375  C9          ret

----- write BACKGROUND/FOREGROUND
      @ 1362!
1376  2A 91 B2    ld hl,(B291)     TXT address of BACK/FOREGROUND routine
1379  E9          jp (hl)
```

```
----- TXT SET BACKground being written <a>
      @ 114F! 14E5  BB9F!
137A  21 91 13     ld hl,1391       FOREGROUND
137D  B7           or a
137E  28 03        jr z,1383        skip if zero (=FOREGROUND)
1380  21 9F 13     ld hl,139F       BACKGROUND
1383  22 91 B2     ld (B291),hl     TXT address of BACK/FOREGROUND routine
1386  C9           ret

----- TXT GET if BACKground is being written <a>
      @ BBA2!
1387  2A 91 B2     ld hl,(B291)     TXT address of BACK/FOREGROUND routine
138A  11 6F EC     ld de,EC6F
138D  19           add hl,de
138E  7C           ld a,h
138F  B5           or 1
1390  C9           ret

----- FOREGROUND
      @ 137A: B291:
1391  2A 8F B2     ld hl,(B28F)     TXT PEN ink
1394  79           ld a,c
1395  2F           cpl
1396  A4           and h
1397  47           ld b,a
1398  79           ld a,c
1399  A5           and 1
139A  B0           or b
139B  0E FF        ld c,FF          =255.
139D  18 03        jr 13A2

----- BACKGROUND
      @ 1380:
139F  3A 8F B2     ld a,(B28F)      TXT PEN ink
13A2  47           ld b,a
13A3  EB           ex de,hl
13A4  C3 6B 0C     jp 0C6B          SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl

----- TXT SET GRAPHIC char write, <a>=0=OFF, FF=ON
      @ 114C! BB63!
13A7  32 93 B2     ld (B293),a      TXT flag graphic char write
13AA  C9           ret

----- TXT READ char from screen <hl>=col/row, =<a>, =carry
      @ 2D09! BB60!
13AB  E5           push hl
13AC  D5            push de
13AD  C5             push bc
13AE  CD D0 BD        call BDD0     TXT DRAW/UNDRAW CURSOR, if enabled
13B1  2A 85 B2        ld hl,(B285)  TXT CURSOR column/row
13B4  CD D6 BD        call BDD6     TXT UNWRITE CHAR, read screen <hl>=col/row,
13B7  F5              push af
13B8  CD CD BD         call BDCD    TXT DRAW/UNDRAW CURSOR, if enabled
13BB  F1              pop af
13BC  C1             pop bc
13BD  D1            pop de
13BE  E1           pop hl
13BF  C9           ret

----- TXT UNWRITE CHAR, read screen <hl>=col/row, =<a>
      @ BDD6
13C0  3A 8F B2     ld a,(B28F)      TXT PEN ink
13C3  11 98 B2     ld de,B298       TXT buffer for unpacked char matrix
13C6  E5           push hl
13C7  D5            push de
```

```
13C8  CD 49 0F      call 0F49      SCR REPACK char matrix, <a>=ink to match, (h
13CB  CD E3 13      call 13E3      compare screen matrix with matrix table
13CE  D1            pop de
13CF  E1            pop hl
13D0  30 01         jr nc,13D3     ...
13D2  C0            ret nz
13D3  3A 90 B2      ld a,(B290)    TXT PAPER ink
13D6  D5            push de
13D7  CD 49 0F      call 0F49      SCR REPACK char matrix, <a>=ink to match, (h
13DA  D1            pop de
13DB  06 08         ld b,08        =8.
13DD  1A            ld a,(de)
13DE  2F            cpl
13DF  12            ld (de),a
13E0  13            inc de
13E1  10 FA         djnz 13DD      next

----- compare screen matrix with matrix table
      @ 13CB!
13E3  0E 00         ld c,00        start with matrix of char 0
13E5  79            ld a,c
13E6  CD D3 12      call 12D3      TXT GET char <a> MATRIX, (hl)=address, carry
13E9  11 98 B2      ld de,B298     TXT buffer for unpacked char matrix
13EC  06 08         ld b,08        # of bytes
13EE  1A            ld a,(de)
13EF  BE            cp (hl)
13F0  20 09         jr nz,13FB     no match
13F2  23            inc hl
13F3  13            inc de
13F4  10 F8         djnz 13EE      next byte of matrix
13F6  79            ld a,c
13F7  FE 20         cp 20          'SPACE
13F9  37            scf
13FA  C9            ret

13FB  0C            inc c
13FC  20 E7         jr nz,13E5     try next byte of matrix
13FE  AF            xor a
13FF  C9            ret

----- TXT OUTPUT char or ctl code <a> to VDU
      @ 06EF! 2780  2B3D! 2B59! 2B5E  2D60! BB5A!
1400  F5            push af
1401  C5             push bc
1402  D5              push de
1403  E5               push hl
1404  CD D9 BD          call BDD9  TXT OUT ACTION, char or ctl code <a> to VDU
1407  E1               pop hl
1408  D1              pop de
1409  C1             pop bc
140A  F1            pop af
140B  C9            ret

----- TXT OUT ACTION, char or ctl code <a> to VDU
      @ BDD9
140C  4F            ld c,a
140D  3A 93 B2      ld a,(B293)    TXT flag graphic char write
1410  B7            or a
1411  79            ld a,c
1412  C2 45 19      jp nz,1945     GRA WRITE CHAR <a> at current graphic pos
1415  21 B8 B2      ld hl,B2B8     TXT control code buffer index
1418  46            ld b,(hl)
1419  78            ld a,b
141A  FE 0A         cp 0A          more then 9 parameters?
141C  30 28         jr nc,1446     yes, that's wrong, reset index
```

```
141E  B7          or a              are there control code parameters?
141F  20 06       jr nz,1427        proceed with parameters
1421  79          ld a,c            is it a normal character?
1422  FE 20       cp 20             'SPACE
1424  D2 34 13    jp nc,1334        TXT WRITE char <a> to screen

----- proceed with parameters
1427  04          inc b
1428  70          ld (hl),b
1429  58          ld e,b
142A  16 00       ld d,00           clear hi part
142C  19          add hl,de
142D  71          ld (hl),c
142E  3A B9 B2    ld a,(B2B9)       TXT control code buffer for up to 9 paramete
1431  5F          ld e,a
1432  21 C3 B2    ld hl,B2C3        control code table; <# of parameters>, <rout
1435  19          add hl,de
1436  19          add hl,de
1437  19          add hl,de         calculate entry to control code table
1438  7E          ld a,(hl)         get # of parameters
1439  B8          cp b
143A  D0          ret nc
143B  23          inc hl
143C  5E          ld e,(hl)
143D  23          inc hl
143E  56          ld d,(hl)
143F  21 B9 B2    ld hl,B2B9        TXT control code buffer for up to 9 paramete
1442  79          ld a,c
1443  CD 16 00    call 0016         jp(de)
1446  AF          xor a             reset
1447  32 B8 B2    ld (B2B8),a       TXT control code buffer index
144A  C9          ret

----- TXT VDU DISABLE
      @ BB57!
144B  CD 9A 12    call 129A         TXT CURSOR DISABLE (user)
144E  AF          xor a
144F  18 05       jr 1456

----- TXT VDU ENABLE
      @ 115B  BB54!
1451  CD 89 12    call 1289         TXT CURSOR ENABLE (user)
1454  3E FF       ld a,FF           'IGNORE
1456  32 8E B2    ld (B28E),a       TXT flag VDU enable
1459  18 EB       jr 1446           reset control code buffer index

----- TXT reset to default control code table
      @ 108E
145B  AF          xor a
145C  32 B8 B2    ld (B2B8),a       TXT control code buffer index
145F  21 6B 14    ld hl,146B        data for control code table (copied to B2C3)
1462  11 C3 B2    ld de,B2C3        control code table; <# of parameters>, <rout
1465  01 60 00    ld bc,0060        byte count of table  (copied to B2C3)
1468  ED B0       ldir
146A  C9          ret

----- data for control code table (copied to B2C3)
      @ 145F:
146B  00 E2 14 01 34 13 00 9A  12 00 89 12 01 CA 0A 01  45 19 00 51 14 00 D8 14
1483  00 0A 15 00 0F 15 00 14  15 00 19 15 00 40 15 00  30 15 01 AE 12 01 A9 12
149B  00 4F 15 00 8E 15 00 84  15 00 6D 15 00 56 15 00  4B 14 01 E3 14 01 49 0C
14B3  00 C9 12 09 04 15 04 F8  14 00 E2 14 03 E8 14 02  F1 14 00 2A 15 02 38 15
```

```
----- TXT GET CONTROL code table addr
      @ BBB1!
14CB  21 C3 B2     ld hl,B2C3         control code table; <# of parameters>, <rout
14CE  C9           ret

----- data to ring 'BELL
      @ 14DA:
14CF  87 00 00 5A 00 00 0B 14  00

----- TXT ring the bell
14D8  DD E5        push ix
14DA  21 CF 14      ld hl,14CF        data to ring 'BELL
14DD  CD 9F 1F      call 1F9F         SOUND QUEUE, add a sound, (hl)=sound program
14E0  DD E1        pop ix
14E2  C9           ret

----- TXT set write mode 2, 0=off, 1=on
      @ B305
14E3  0F           rrca
14E4  9F           sbc a,a
14E5  C3 7A 13     jp 137A            TXT SET BACKground being written <a>

----- TXT set ink, <PEN>,<colour1>,[<colour2>]
      @ B317
14E8  23           inc hl
14E9  7E           ld a,(hl)
14EA  23           inc hl
14EB  46           ld b,(hl)
14EC  23           inc hl
14ED  4E           ld c,(hl)
14EE  C3 EC 0C     jp 0CEC            SCR SET colour of INK, <a>=ink#, <b,c>=colou

----- TXT set border <colour>[<colour>]
      @ B31A
14F1  23           inc hl
14F2  46           ld b,(hl)
14F3  23           inc hl
14F4  4E           ld c,(hl)
14F5  C3 F1 0C     jp 0CF1            SCR SET BORDER, <b,c>=colours

----- TXT define window, <left>,<right>,<top>,<bottom>
      @ B311
14F8  23           inc hl
14F9  56           ld d,(hl)
14FA  23           inc hl
14FB  7E           ld a,(hl)
14FC  23           inc hl
14FD  5E           ld e,(hl)
14FE  23           inc hl
14FF  6E           ld l,(hl)
1500  67           ld h,a
1501  C3 0C 12     jp 120C            TXT SET WINDOW <hl>=left top, <de>=right bot

----- TXT set matrix for user <symbol>, 8<byte matrix>
      @30E
1504  23           inc hl
1505  7E           ld a,(hl)
1506  23           inc hl
1507  C3 F1 12     jp 12F1            TXT SET char MATRIX, <a>=char, (hl)=matrix t

----- TXT cursor left one step
      @ B2DB
150A  11 00 FF     ld de,FF00         <d>=-1 cursor horizontal, <e>=0 vertical
150D  18 0D        jr 151C
```

```
----- TXT cursor right one step
      @ B2DE
150F  11 00 01    ld de,0100        horizontal +1
1512  18 08       jr 151C

----- TXT cursor down one line
      @ B2E1
1514  11 01 00    ld de,0001        vertical +1
1517  18 03       jr 151C

----- TXT cursor up one line
      @ B2E4
1519  11 FF 00    ld de,00FF        vertical -1
151C  D5          push de
151D  CD A8 11     call 11A8        remove cursor, validate, scroll up
1520  D1          pop de
1521  7D          ld a,l
1522  83          add a,e
1523  6F          ld l,a
1524  7C          ld a,h
1525  82          add a,d
1526  67          ld h,a
1527  C3 7A 11    jp 117A           draw cursor, if enabled

----- TXT cursor HOME
      @ B31D
152A  2A 88 B2    ld hl,(B288)      TXT row; window left upper corner
152D  C3 77 11    jp 1177           undraw cursor, if enabled

----- TXT cursor to start of line
      @ B2ED
1530  CD A8 11    call 11A8         remove cursor, validate, scroll up
1533  3A 89 B2    ld a,(B289)       TXT column, window left upper corner
1536  18 EE       jr 1526           set column to start of window

----- TXT cursor LOCATE <column>(de),<line>(de+1)
      @ B320
1538  23          inc hl
1539  56          ld d,(hl)
153A  23          inc hl
153B  5E          ld e,(hl)
153C  EB          ex de,hl
153D  C3 74 11    jp 1174           TXT SET CURSOR, <hl>=column/row

----- TXT CLEAR current WINDOW
      @ BB6C!
1540  CD D0 BD    call BDD0         TXT DRAW/UNDRAW CURSOR, if enabled
1543  2A 88 B2    ld hl,(B288)      TXT row; window left upper corner
1546  22 85 B2    ld (B285),hl      TXT CURSOR column/row
1549  ED 5B 8A B2 ld de,(B28A)      TXT row, window right bottom corner
154D  18 48       jr 1597           clear this window

----- TXT clear char at cursor position
      @ B2F3
154F  CD A8 11    call 11A8         remove cursor, validate, scroll up
1552  54          ld d,h
1553  5D          ld e,l
1554  18 41       jr 1597           clear this window

----- TXT clear window from cursor to end
      @ B2FF
1556  CD 84 15    call 1584         TXT clear line from cursor to end
1559  2A 88 B2    ld hl,(B288)      TXT row; window left upper corner
155C  ED 5B 8A B2 ld de,(B28A)      TXT row, window right bottom corner
1560  3A 85 B2    ld a,(B285)       TXT CURSOR column/row
```

```
1563  6F          ld l,a
1564  2C          inc l
1565  BB          cp e
1566  3A 90 B2    ld a,(B290)       TXT PAPER ink
1569  DC B3 0D    call c,0DB3       SCR FILL BOX, <a>=ink, <hl,de>=corners
156C  C9          ret

----- TXT clear window from start to cursor
      @ B2F6
156D  CD 8E 15    call 158E         TXT clear start of line incl cursor
1570  2A 88 B2    ld hl,(B288)      TXT row; window left upper corner
1573  3A 8B B2    ld a,(B28B)       TXT column, window right bottom corner
1576  57          ld d,a
1577  3A 85 B2    ld a,(B285)       TXT CURSOR column/row
157A  3D          dec a
157B  5F          ld e,a
157C  BD          cp l
157D  3A 90 B2    ld a,(B290)       TXT PAPER ink
1580  D4 B3 0D    call nc,0DB3      SCR FILL BOX, <a>=ink, <hl,de>=corners
1583  C9          ret

----- TXT clear line from cursor to end
      @ 1556! B2F9
1584  CD A8 11    call 11A8         remove cursor, validate, scroll up
1587  5D          ld e,l
1588  3A 8B B2    ld a,(B28B)       TXT column, window right bottom corner
158B  57          ld d,a
158C  18 09       jr 1597           clear this window

----- TXT clear start of line incl cursor
      @ 156D! B2F6
158E  CD A8 11    call 11A8         remove cursor, validate, scroll up
1591  EB          ex de,hl
1592  6B          ld l,e
1593  3A 89 B2    ld a,(B289)       TXT column, window left upper corner
1596  67          ld h,a

----- clear this window
1597  3A 90 B2    ld a,(B290)       TXT PAPER ink
159A  CD B3 0D    call 0DB3         SCR FILL BOX, <a>=ink, <hl,de>=corners
159D  CD CD BD    call BDCD         TXT DRAW/UNDRAW CURSOR, if enabled
15A0  C9          ret

15A1  C7 C7 C7 C7 C7 C7 C7 C7  C7 C7 C7 C7 C7 C7 C7
```

```
----- GRA INITIALISE graphics VDU
      @ 0649! BBBA!
15B0  CD DF 15    call 15DF         GRA RESET
15B3  21 01 00    ld hl,0001        PAPER=0, PEN=1

      @ 0AE5!
15B6  7C          ld a,h
15B7  CD FD 17    call 17FD         GRA SET PAPER, <a>=ink
15BA  7D          ld a,l
15BB  CD F6 17    call 17F6         GRA SET PEN, <a>=ink
15BE  21 00 00    ld hl,0000
15C1  54          ld d,h
15C2  5D          ld e,l
15C3  CD 04 16    call 1604         GRA SET ORIGIN, <de>=x, <hl>=y
15C6  11 00 80    ld de,8000
15C9  21 FF 7F    ld hl,7FFF
15CC  E5          push hl
15CD  D5           push de
15CE  CD 34 17      call 1734       GRA set WINDOW width, <de>=xl, <hl>=x2
15D1  E1           pop hl
15D2  D1          pop de
15D3  C3 79 17    jp 1779           GRA set WINDOW height, <de>=yl, <hl>=y2

      @ 0AD9!
15D6  CD 0A 18    call 180A         GRA GET PAPER, <a>=ink
15D9  67          ld h,a
15DA  CD 04 18    call 1804         GRA GET PEN, <a>=ink
15DD  6F          ld l,a
15DE  C9          ret

----- GRA RESET
      @ 15B0! BBBD!
15DF  21 E5 15    ld hl,15E5        data for graphics jumpblock
15E2  C3 8A 0A    jp 0A8A           copy (hl) bytes to address (hl+1),(hl+2)

----- data for graphics jumpblock
15E5  09 DC BD C3 16 18 C3 2A  18 C3 3C 18

----- GRA MOVE RELATIVE, <de>=xd, <hl>=yd
15F1  CD 57 16    call 1657         add de/hl to graphic cursor

----- GRA MOVE ABSOLUTE, <de>=x, <hl>=y
      @ 161D! BBC0!
15F4  ED 53 2C B3 ld (B32C),de      GRA cursor x
15F8  22 2E B3    ld (B32E),hl      GRA cursor y
15FB  C9          ret

----- GRA ASK CURSOR, <de>=x, <hl>=y
      @ 161A! BBC6!
15FC  ED 5B 2C B3 ld de,(B32C)      GRA cursor x
1600  2A 2E B3    ld hl,(B32E)      GRA cursor y
1603  C9          ret

----- GRA SET ORIGIN, <de>=x, <hl>=y
      @ BBC9!
1604  ED 53 28 B3 ld (B328),de      GRA user origin x
1608  22 2A B3    ld (B32A),hl      GRA user origin y

----- GRA cursor HOME, x=y=0000
      @ 17F3
160B  11 00 00    ld de,0000
160E  62          ld h,d
160F  6B          ld l,e            hl=0000
1610  18 E2       jr 15F4           GRA MOVE ABSOLUTE, <de>=x, <hl>=y
```

```
----- GRA GET ORIGIN <de>=x, <hl>=y of user coordinates
      @ BBCC!
1612  ED 5B 28 B3 ld de,(B328)     GRA user origin x
1616  2A 2A B3    ld hl,(B32A)     GRA user origin y
1619  C9          ret

      @ 183E! 1955!
161A  CD FC 15    call 15FC        GRA ASK CURSOR, <de>=x, <hl>=y

      @ 16FC! 184A!
161D  CD F4 15    call 15F4        GRA MOVE ABSOLUTE, <de>=x, <hl>=y
1620  E5          push hl
1621  CD EC 0A     call 0AEC       SCR GET MODE <a>, cp 01
1624  2F           cpl
1625  C6 01        add a,01        =1.
1627  CE 02        adc a,02        =2.
1629  26 00        ld h,00         =0.
162B  6F           ld l,a
162C  CB 7A        bit 7,d
162E  28 03        jr z,1633       ...
1630  EB           ex de,hl
1631  19           add hl,de
1632  EB           ex de,hl
1633  2F           cpl
1634  A3           and e
1635  5F           ld e,a
1636  7D           ld a,l
1637  2A 28 B3     ld hl,(B328)    GRA user origin x
163A  19           add hl,de
163B  0F           rrca
163C  DC 74 17     call c,1774     sra h; rr l; ret
163F  0F           rrca
1640  DC 74 17     call c,1774     sra h; rr l; ret
1643  D1          pop de
1644  E5          push hl
1645  7A           ld a,d
1646  07           rlca
1647  30 01        jr nc,164A      ...
1649  13           inc de
164A  7B           ld a,e
164B  E6 FE        and FE          =254.
164D  5F           ld e,a
164E  2A 2A B3     ld hl,(B32A)    GRA user origin y
1651  19           add hl,de
1652  CD 74 17     call 1774       sra h; rr l; ret
1655  D1          pop de
1656  C9          ret

----- add de/hl to graphic cursor
      @ 15F1! 1810! 1824! 1836!
1657  E5          push hl
1658  2A 2C B3     ld hl,(B32C)    GRA cursor x
165B  19           add hl,de
165C  D1          pop de
165D  E5          push hl
165E  2A 2E B3     ld hl,(B32E)    GRA cursor y
1661  19           add hl,de
1662  D1          pop de
1663  C9          ret

      @ 192C!
1664  D5          push de
1665  E5           push hl
1666  2A 30 B3      ld hl,(B330)   GRA WINDOW WIDTH, xleft
1669  2B            dec hl
```

```
166A  B7           or a
166B  ED 52        sbc hl,de
166D  F2 AC 16     jp p,16AC       ...
1670  2A 32 B3     ld hl,(B332)    GRA WINDOW WIDTH, xright
1673  B7           or a
1674  ED 52        sbc hl,de
1676  FA AC 16     jp m,16AC       ...
1679  D1          pop de
167A  2A 34 B3    ld hl,(B334)     GRA WINDOW HEIGHT, ytop
167D  B7          or a
167E  ED 52       sbc hl,de
1680  FA AD 16    jp m,16AD        ...
1683  2A 36 B3    ld hl,(B336)     GRA WINDOW HEIGHT, ybottom
1686  2B          dec hl
1687  B7          or a
1688  ED 52       sbc hl,de
168A  FA 91 16    jp m,1691        ...
168D  ED 5B 36 B3 ld de,(B336)     GRA WINDOW HEIGHT, ybottom
1691  2A 36 B3    ld hl,(B336)     GRA WINDOW HEIGHT, ybottom
1694  2B          dec hl
1695  B7          or a
1696  ED 42       sbc hl,bc
1698  F2 AD 16    jp p,16AD        ...
169B  2A 34 B3    ld hl,(B334)     GRA WINDOW HEIGHT, ytop
169E  B7          or a
169F  ED 42       sbc hl,bc
16A1  F2 A8 16    jp p,16A8        ...
16A4  ED 4B 34 B3 ld bc,(B334)     GRA WINDOW HEIGHT, ytop
16A8  EB          ex de,hl
16A9  D1         pop de
16AA  37         scf
16AB  C9         ret

16AC  E1          pop hl
16AD  D1         pop de
16AE  B7         or a
16AF  C9         ret

      @ 1907!
16B0  E5         push hl
16B1  D5          push de
16B2  EB            ex de,hl
16B3  2A 36 B3      ld hl,(B336)   GRA WINDOW HEIGHT, ybottom
16B6  2B            dec hl
16B7  B7            or a
16B8  ED 52         sbc hl,de
16BA  F2 F8 16      jp p,16F8      ...
16BD  2A 34 B3      ld hl,(B334)   GRA WINDOW HEIGHT, ytop
16C0  B7            or a
16C1  ED 52         sbc hl,de
16C3  FA F8 16      jp m,16F8      ...
16C6  D1          pop de
16C7  2A 32 B3    ld hl,(B332)     GRA WINDOW WIDTH, xright
16CA  B7          or a
16CB  ED 52       sbc hl,de
16CD  FA F9 16    jp m,16F9        ...
16D0  2A 30 B3    ld hl,(B330)     GRA WINDOW WIDTH, xleft
16D3  2B          dec hl
16D4  B7          or a
16D5  ED 52       sbc hl,de
16D7  FA DE 16    jp m,16DE        ...
16DA  ED 5B 30 B3 ld de,(B330)     GRA WINDOW WIDTH, xleft
16DE  2A 30 B3    ld hl,(B330)     GRA WINDOW WIDTH, xleft
16E1  2B          dec hl
16E2  B7          or a
```

```
16E3  ED 42          sbc hl,bc
16E5  F2 F9 16       jp p,16F9        ...
16E8  2A 32 B3       ld hl,(B332)     GRA WINDOW WIDTH, xright
16EB  B7             or a
16EC  ED 42          sbc hl,bc
16EE  F2 F5 16       jp p,16F5        ...
16F1  ED 4B 32 B3    ld bc,(B332)     GRA WINDOW WIDTH, xright
16F5  E1             pop hl
16F6  37             scf
16F7  C9             ret

16F8  D1             pop de
16F9  E1             pop hl
16FA  B7             or a
16FB  C9             ret

----- check if point is inside GRA WINDOW
      @ 1816! 182A!
16FC  CD 1D 16       call 161D        ...

      @ 1958! 1968! 19AE!
16FF  E5             push hl
1700  2A 30 B3       ld hl,(B330)     GRA WINDOW WIDTH, xleft
1703  2B             dec hl
1704  B7             or a
1705  ED 52          sbc hl,de
1707  F2 2D 17       jp p,172D        ...
170A  2A 32 B3       ld hl,(B332)     GRA WINDOW WIDTH, xright
170D  B7             or a
170E  ED 52          sbc hl,de
1710  FA 2D 17       jp m,172D        ...
1713  E1             pop hl
1714  D5             push de
1715  EB             ex de,hl
1716  2A 36 B3       ld hl,(B336)     GRA WINDOW HEIGHT, ybottom
1719  2B             dec hl
171A  B7             or a
171B  ED 52          sbc hl,de
171D  F2 30 17       jp p,1730        ...
1720  2A 34 B3       ld hl,(B334)     GRA WINDOW HEIGHT, ytop
1723  B7             or a
1724  ED 52          sbc hl,de
1726  FA 30 17       jp m,1730        ...
1729  EB             ex de,hl
172A  D1             pop de
172B  37             scf
172C  C9             ret

172D  E1             pop hl
172E  B7             or a
172F  C9             ret

1730  EB             ex de,hl
1731  D1             pop de
1732  B7             or a
1733  C9             ret

----- GRA set WINDOW width, <de>=x1, <hl>=x2
      @ 15CE! BBCF!
1734  E5             push hl
1735  CD 60 17       call 1760        ...
1738  D1             pop de
1739  E5             push hl
173A  CD 60 17       call 1760        ...
173D  D1             pop de
```

```
173E  7B          ld a,e
173F  95          sub l
1740  7A          ld a,d
1741  9C          sbc a,h
1742  38 01       jr c,1745        ...
1744  EB          ex de,hl
1745  7B          ld a,e
1746  E6 F8       and F8           =248.
1748  5F          ld e,a
1749  7D          ld a,l
174A  F6 07       or 07            =7.
174C  6F          ld l,a
174D  CD EC 0A    call 0AEC        SCR GET MODE <a>, cp 01
1750  3D          dec a
1751  FC 70 17    call m,1770      sra d; rr e; sra h; rr l; ret
1754  3D          dec a
1755  FC 70 17    call m,1770      sra d; rr e; sra h; rr l; ret
1758  ED 53 30 B3 ld (B330),de     GRA WINDOW WIDTH, xleft
175C  22 32 B3    ld (B332),hl     GRA WINDOW WIDTH, xright
175F  C9          ret

      @ 1735! 173A!
1760  7A          ld a,d
1761  B7          or a
1762  21 00 00    ld hl,0000
1765  F8          ret m
1766  21 7F 02    ld hl,027F       ...
1769  7B          ld a,e
176A  95          sub l
176B  7A          ld a,d
176C  9C          sbc a,h
176D  D0          ret nc
176E  EB          ex de,hl
176F  C9          ret

----- sra d; rr e; sra h; rr l; ret
      @ 1751! 1755!
1770  CB 2A       sra d
1772  CB 1B       rr e

----- sra h; rr l; ret
      @ 163C! 1640! 1652! 17CC! 17CF!
1774  CB 2C       sra h
1776  CB 1D       rr l
1778  C9          ret

----- GRA set WINDOW height, <de>=y1, <hl>=y2
      @ 15D3  BBD2!
1779  E5          push hl
177A  CD 92 17     call 1792       ...
177D  D1          pop de
177E  E5          push hl
177F  CD 92 17     call 1792       ...
1782  D1          pop de
1783  7D          ld a,l
1784  93          sub e
1785  7C          ld a,h
1786  9A          sbc a,d
1787  38 01       jr c,178A        ...
1789  EB          ex de,hl
178A  ED 53 34 B3 ld (B334),de     GRA WINDOW HEIGHT, ytop
178E  22 36 B3    ld (B336),hl     GRA WINDOW HEIGHT, ybottom
1791  C9          ret
```

```
1792  7A          ld a,d
1793  B7          or a
1794  21 00 00    ld hl,0000
1797  F8          ret m
1798  CB 3A       srl d
179A  CB 1B       rr e
179C  21 C7 00    ld hl,00C7        ...
179F  7B          ld a,e
17A0  95          sub l
17A1  7A          ld a,d
17A2  9C          sbc a,h
17A3  D0          ret nc
17A4  EB          ex de,hl
17A5  C9          ret

----- GRA get WINDOW width, <de>=xleft, <hl>=xright>
      @ 17C5! BBD5!
17A6  ED 5B 30 B3 ld de,(B330)      GRA WINDOW WIDTH, xleft
17AA  2A 32 B3    ld hl,(B332)      GRA WINDOW WIDTH, xright
17AD  CD EC 0A    call 0AEC         SCR GET MODE <a>, cp 01
17B0  3D          dec a
17B1  FC B6 17    call m,17B6       mode was 0
17B4  3D          dec a
17B5  F0          ret p
17B6  29          add hl,hl
17B7  23          inc hl
17B8  EB          ex de,hl
17B9  29          add hl,hl
17BA  EB          ex de,hl
17BB  C9          ret

----- GRA GET WINDOW HEIGHT, <de>=ytop, <hl>=ybottom
      @ BBD8!
17BC  ED 5B 34 B3 ld de,(B334)      GRA WINDOW HEIGHT, ytop
17C0  2A 36 B3    ld hl,(B336)      GRA WINDOW HEIGHT, ybottom
17C3  18 F1       jr 17B6

----- GRA CLEAR GRAPHIC WINDOW
      @ BBDB!
17C5  CD A6 17    call 17A6         GRA get WINDOW width, <de>=xleft, <hl>=xrigh
17C8  B7          or a
17C9  ED 52       sbc hl,de
17CB  23          inc hl
17CC  CD 74 17    call 1774         sra h; rr l; ret
17CF  CD 74 17    call 1774         sra h; rr l; ret
17D2  CB 3D       srl l
17D4  45          ld b,l
17D5  ED 5B 36 B3 ld de,(B336)      GRA WINDOW HEIGHT, ybottom
17D9  2A 34 B3    ld hl,(B334)      GRA WINDOW HEIGHT, ytop
17DC  E5          push hl
17DD  B7           or a
17DE  ED 52        sbc hl,de
17E0  23           inc hl
17E1  4D           ld c,l
17E2  ED 5B 30 B3  ld de,(B330)     GRA WINDOW WIDTH, xleft
17E6  E1          pop hl
17E7  C5          push bc
17E8  CD A9 0B     call 0BA9        SCR DOT POSITION convert base coordinates to
17EB  D1          pop de
17EC  3A 39 B3    ld a,(B339)       GRA PAPER ink
17EF  4F          ld c,a
17F0  CD B7 0D    call 0DB7         SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=
17F3  C3 0B 16    jp 160B           GRA cursor HOME, x=y=0000
```

```
----- GRA SET PEN, <a>=ink
      @ 15BB! BBDE!
17F6  CD 86 0C     call 0C86        SCR INK ENCODE, in: <a>=ink#, out: <a>=encod
17F9  32 38 B3     ld (B338),a      GRA PEN INK
17FC  C9           ret

----- GRA SET PAPER, <a>=ink
      @ 15B7! BBE4!
17FD  CD 86 0C     call 0C86        SCR INK ENCODE, in: <a>=ink#, out: <a>=encod
1800  32 39 B3     ld (B339),a      GRA PAPER ink
1803  C9           ret

----- GRA GET PEN, <a>=ink
      @ 15DA! BBE1!
1804  3A 38 B3     ld a,(B338)      GRA PEN INK
1807  C3 A0 0C     jp 0CA0          SCR INK DECODE, in: <a>=encoded ink; out: <a

----- GRA GET PAPER, <a>=ink
      @ 15D6! 182D  BBE7!
180A  3A 39 B3     ld a,(B339)      GRA PAPER ink
180D  C3 A0 0C     jp 0CA0          SCR INK DECODE, in: <a>=encoded ink; out: <a

----- GRA PLOT RELATIVE, <de>=xd, <hl>=yd
      @ BBED!
1810  CD 57 16     call 1657        add de/hl to graphic cursor

----- GRA PLOT ABSOLUTE, <de>=x, <hl>=y
      @ BBEA!
1813  C3 DC BD     jp BDDC          GRA PLOT a POINT, <de>=x, <hl>=y

----- GRA PLOT a POINT, <de>=x, <hl>=y
      @ BDDC
1816  CD FC 16     call 16FC        check if point is inside GRA WINDOW
1819  D0           ret nc
181A  CD A9 0B     call 0BA9        SCR DOT POSITION convert base coordinates to
181D  3A 38 B3     ld a,(B338)      GRA PEN INK
1820  47           ld b,a
1821  C3 E8 BD     jp BDE8          SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin

----- GRA TEST RELATIVE, <de>=xd, <hl>=yd
      @ BBF3!
1824  CD 57 16     call 1657        add de/hl to graphic cursor

----- GRA TEST ABSOLUTE, <de>=x, <hl>=y
      @ BBF0!
1827  C3 DF BD     jp BDDF          GRA TEST a POINT, <de>=x, <hl>=y

----- GRA TEST a POINT, <de>=x, <hl>=y
      @ BDDF
182A  CD FC 16     call 16FC        check if point is inside GRA WINDOW
182D  D2 0A 18     jp nc,180A       GRA GET PAPER, <a>=ink
1830  CD A9 0B     call 0BA9        SCR DOT POSITION convert base coordinates to
1833  C3 E5 BD     jp BDE5          SCR READ a pixel from the screen, (hl)=addr,

----- GRA DRAW LINE RELATIVE, <de>=xd, <hl>=yd
      @ BBF9!
1836  CD 57 16     call 1657        add de/hl to graphic cursor

----- GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
      @ BBF6!
1839  C3 E2 BD     jp BDE2          GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
```

```
----- GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
      @ BDE2
183C  E5            push hl
183D  D5             push de
183E  CD 1A 16        call 161A       ...
1841  ED 53 42 B3     ld (B342),de    GRA temp store x on draw
1845  22 44 B3        ld (B344),hl    GRA temp store y on draw
1848  D1             pop de
1849  E1            pop hl
184A  CD 1D 16      call 161D         ...
184D  E5            push hl
184E  2A 42 B3       ld hl,(B342)     GRA temp store x on draw
1851  B7             or a
1852  ED 52          sbc hl,de
1854  44             ld b,h
1855  4D             ld c,l
1856  FA 69 18       jp m,1869        ...
1859  2A 42 B3       ld hl,(B342)     GRA temp store x on draw
185C  EB             ex de,hl
185D  22 42 B3       ld (B342),hl     GRA temp store x on draw
1860  2A 44 B3       ld hl,(B344)     GRA temp store y on draw
1863  E3             ex (sp),hl
1864  22 44 B3       ld (B344),hl     GRA temp store y on draw
1867  18 08          jr 1871          ...

1869  21 00 00       ld hl,0000
186C  B7             or a
186D  ED 42          sbc hl,bc
186F  44             ld b,h
1870  4D             ld c,l
1871  D1            pop de
1872  2A 44 B3      ld hl,(B344)      GRA temp store y on draw
1875  B7            or a
1876  ED 52         sbc hl,de
1878  EB            ex de,hl
1879  F2 8E 18      jp p,188E         ...
187C  21 00 00      ld hl,0000
187F  B7            or a
1880  ED 52         sbc hl,de
1882  54            ld d,h
1883  5D            ld e,l
1884  B7            or a
1885  ED 42         sbc hl,bc
1887  21 01 00      ld hl,0001        ...
188A  30 27         jr nc,18B3        ...
188C  18 0A         jr 1898           ...

188E  62            ld h,d
188F  6B            ld l,e
1890  B7            or a
1891  ED 42         sbc hl,bc
1893  21 FF FF      ld hl,FFFF        ...
1896  30 09         jr nc,18A1        ...
1898  22 3A B3      ld (B33A),hl      GRA temp store 1
189B  60            ld h,b
189C  69            ld l,c
189D  3E FF         ld a,FF           =255.
189F  18 19         jr 18BA           ...

18A1  E5            push hl
18A2  2A 42 B3       ld hl,(B342)     GRA temp store x on draw
18A5  09             add hl,bc
18A6  22 42 B3       ld (B342),hl     GRA temp store x on draw
18A9  2A 44 B3       ld hl,(B344)     GRA temp store y on draw
18AC  B7             or a
```

```
18AD  ED 52         sbc hl,de
18AF  22 44 B3      ld (B344),hl     GRA temp store y on draw
18B2  E1           pop hl
18B3  22 3A B3     ld (B33A),hl     GRA temp store 1

      @ 17B1! 17C3'
18B6  60           ld h,b
18B7  69           ld l,c
18B8  EB           ex de,hl
18B9  AF           xor a            reset
18BA  32 46 B3     ld (B346),a      GRA temp flag
18BD  13           inc de
18BE  ED 53 40 B3 ld (B340),de     GRA temp store 4
18C2  23           inc hl
18C3  CD 8C 37     call 378C        INT ARITH, DVDu <hl>=<hl>/<de>; <de>=remaind
18C6  22 3C B3     ld (B33C),hl     GRA temp store 2
18C9  ED 53 3E B3 ld (B33E),de     GRA temp store 3
18CD  ED 4B 40 B3 ld bc,(B340)     GRA temp store 4
18D1  50           ld d,b
18D2  59           ld e,c
18D3  CB 3A        srl d
18D5  CB 1B        rr e
18D7  C5           push bc
18D8  ED 4B 3C B3   ld bc,(B33C)    GRA temp store 2
18DC  2A 3E B3      ld hl,(B33E)    GRA temp store 3
18DF  19            add hl,de
18E0  EB            ex de,hl
18E1  2A 40 B3      ld hl,(B340)    GRA temp store 4
18E4  B7            or a
18E5  ED 52         sbc hl,de
18E7  30 07         jr nc,18F0      ...
18E9  19            add hl,de
18EA  EB            ex de,hl
18EB  B7            or a
18EC  ED 52         sbc hl,de
18EE  EB            ex de,hl
18EF  03            inc bc
18F0  D5            push de
18F1  3A 46 B3       ld a,(B346)    GRA temp flag
18F4  B7             or a
18F5  28 23          jr z,191A      ...
18F7  2A 42 B3       ld hl,(B342)   GRA temp store x on draw
18FA  54             ld d,h
18FB  5D             ld e,l
18FC  09             add hl,bc
18FD  22 42 B3       ld (B342),hl   GRA temp store x on draw
1900  44             ld b,h
1901  4D             ld c,l
1902  0B             dec bc
1903  2A 44 B3       ld hl,(B344)   GRA temp store y on draw
1906  E5             push hl
1907  CD B0 16        call 16B0     ...
190A  3A 38 B3        ld a,(B338)   GRA PEN INK
190D  DC C4 0F        call c,0FC4   SCR HORIZONTAL line plot, <a>=ink, de=xbase,
1910  D1             pop de
1911  2A 3A B3       ld hl,(B33A)   GRA temp store 1
1914  19             add hl,de
1915  22 44 B3       ld (B344),hl   GRA temp store y on draw
1918  18 23          jr 193D        ...

191A  2A 44 B3       ld hl,(B344)   GRA temp store y on draw
191D  54             ld d,h
191E  5D             ld e,l
191F  09             add hl,bc
1920  22 44 B3       ld (B344),hl   GRA temp store y on draw
```

```
1923  44             ld b,h
1924  4D             ld c,l
1925  0B             dec bc
1926  EB             ex de,hl
1927  ED 5B 42 B3    ld de,(B342)    GRA temp store x on draw
192B  D5             push de
192C  CD 64 16        call 1664      ...
192F  3A 38 B3        ld a,(B338)    GRA PEN INK
1932  DC 2F 10        call c,102F    SCR VERTICAL line plot, <a>=ink, de=xbase, b
1935  D1             pop de
1936  2A 3A B3        ld hl,(B33A)   GRA temp store 1
1939  19              add hl,de
193A  22 42 B3        ld (B342),hl   GRA temp store x on draw
193D  D1             pop de
193E  C1            pop bc
193F  0B            dec bc
1940  78            ld a,b
1941  B1            or c
1942  20 93         jr nz,18D7       ...
1944  C9            ret

----- GRA WRITE CHAR <a> at current graphic pos
      @ 1412  BBFC!
1945  DD E5         push ix
1947  CD D3 12        call 12D3        TXT GET char <a> MATRIX, (hl)=address, carry
194A  11 3A B3        ld de,B33A       GRA temp store 1
194D  D5              push de
194E  DD E1           pop ix
1950  01 08 00        ld bc,0008
1953  ED B0           ldir
1955  CD 1A 16        call 161A        ...
1958  CD FF 16        call 16FF        ...
195B  30 4C           jr nc,19A9       ...
195D  E5              push hl
195E  D5               push de
195F  01 07 00          ld bc,0007     ...
1962  EB                ex de,hl
1963  09                add hl,bc
1964  EB                ex de,hl
1965  B7                or a
1966  ED 42             sbc hl,bc
1968  CD FF 16          call 16FF      ...
196B  D1               pop de
196C  E1              pop hl
196D  30 3A           jr nc,19A9       ...
196F  CD A9 0B        call 0BA9        SCR DOT POSITION convert base coordinates to
1972  16 08           ld d,08          =8.
1974  E5              push hl
1975  1E 08            ld e,08         =8.
1977  CD CF 19          call 19CF       ...
197A  CB 09            rrc c
197C  DC F9 0B          call c,0BF9     SCR NEXT BYTE, step screen addr (hl) right o
197F  DD CB 00 06      rlc (ix+00)
1983  1D               dec e
1984  20 F1            jr nz,1977      ...
1986  E1              pop hl
1987  CD 13 0C        call 0C13        SCR NEXT LINE, step screen addr (hl) down on
198A  DD 23           inc ix
198C  15              dec d
198D  20 E5           jr nz,1974       ...
198F  DD E1         pop ix
1991  CD FC 15      call 15FC          GRA ASK CURSOR, <de>=x, <hl>=y
1994  EB            ex de,hl
1995  CD EC 0A      call 0AEC          SCR GET MODE <a>, cp 01
1998  01 08 00      ld bc,0008
```

```
199B  FE 01       cp 01            =1.
199D  28 04       jr z,19A3        ...
199F  30 03       jr nc,19A4       ...
19A1  09          add hl,bc
19A2  09          add hl,bc
19A3  09          add hl,bc
19A4  09          add hl,bc
19A5  EB          ex de,hl
19A6  C3 F4 15    jp 15F4          GRA MOVE ABSOLUTE, <de>=x, <hl>=y

19A9  0E 08        ld c,08         =8.
19AB  D5           push de
19AC  06 08         ld b,08        =8.
19AE  CD FF 16      call 16FF      ...
19B1  30 0C         jr nc,19BF     ...
19B3  E5            push hl
19B4  D5             push de
19B5  C5              push bc
19B6  CD A9 0B         call 0BA9   SCR DOT POSITION convert base coordinates to
19B9  CD CF 19         call 19CF   ...
19BC  C1              pop bc
19BD  D1             pop de
19BE  E1            pop hl
19BF  DD CB 00 06   rlc (ix+00)
19C3  13            inc de
19C4  10 E8         djnz 19AE      ...
19C6  D1           pop de
19C7  2B           dec hl
19C8  DD 23        inc ix
19CA  0D           dec c
19CB  20 DE        jr nz,19AB      ...
19CD  18 C0        jr 198F         ...

      @ 1977! 19B9!
19CF  DD CB 00 7E bit 7,(ix+00)
19D3  3A 38 B3    ld a,(B338)      GRA PEN INK
19D6  20 03       jr nz,19DB       ...
19D8  3A 39 B3    ld a,(B339)      GRA PAPER ink
19DB  47          ld b,a
19DC  C3 E8 BD    jp BDE8          SCR WRITE pixel(s) (hl)=addr, <c>=mask, usin

19DF  C7
```

```
----- KM INITIALISE key manager
      @ 063D! BB00!
19E0  21 02 1E    ld hl,1E02        default value for DELAY KEY
19E3  CD 6D 1C    call 1C6D         KM SET DELAY key, <h>=start, <l>=rep. speed
19E6  AF          xor a
19E7  32 0B B5    ld (B50B),a       ...
19EA  67          ld h,a            HL=0000
19EB  6F          ld l,a
19EC  22 E7 B4    ld (B4E7),hl      KM caps lock state
19EF  21 3C B4    ld hl,B43C        KM KEY REPEAT MAP
19F2  11 B0 FF    ld de,FFB0        =80., =# of keys
19F5  22 47 B5    ld (B547),hl      KM repeat key, pointer to table
19F8  19          add hl,de
19F9  22 45 B5    ld (B545),hl      KM translate control entry, pointer
19FC  19          add hl,de
19FD  22 43 B5    ld (B543),hl      KM translate shift entry, pointer
1A00  19          add hl,de
1A01  22 41 B5    ld (B541),hl      KM translate normal entry, pointer
1A04  EB          ex de,hl
1A05  21 69 1D    ld hl,1D69        default KEY normal/shift/control/repeat entr
1A08  01 FA 00    ld bc,00FA        # of bytes to copy
1A0B  ED B0       ldir
1A0D  06 0A       ld b,0A           # of bytes to clear
1A0F  21 EB B4    ld hl,B4EB        KM key state map (marks pressed keys by sett
1A12  36 00       ld (hl),00        clear key state map to zero
1A14  23          inc hl
1A15  10 FB       djnz 1A12         next
1A17  06 0A       ld b,0A           count
1A19  36 FF       ld (hl),FF        clear rest to FF (no key)
1A1B  23          inc hl
1A1C  10 FB       djnz 1A19         next

----- KM RESET key manager
      @ 05F0! BB03!
1A1E  CD ED 1C    call 1CED         performs reset KM, part 1
1A21  CD 75 1A    call 1A75         set KEYBOARD 'put back' to be empty
1A24  11 46 B4    ld de,B446        KM function KEY expansion buffer
1A27  21 98 00    ld hl,0098        =152. = len of buffer
1A2A  CD 81 1A    call 1A81         performs ALLOCATE EXPANSION BUFFER
1A2D  21 36 1A    ld hl,1A36        data for KM test BREAK or RESET
1A30  CD 8A 0A    call 0A8A         copy (hl) bytes to address (hl+1),(hl+2)
1A33  C3 82 1C    jp 1C82           KM DISARM BREAK

----- data for KM test BREAK or RESET
      @ 1A2D:
1A36  03 EE BD C3 2F 1C

----- KM WAIT CHAR from keyboard =<a>
      @ 1A3F' 2DE3  2DF0! BB06!
1A3C  CD 42 1A    call 1A42         KM READ CHAR from keyboard =<a>
1A3F  30 FB       jr nc,1A3C        KM WAIT CHAR from keyboard =<a>
1A41  C9          ret

----- KM READ CHAR from keyboard =<a>
      @ 1A3C! 2769! BB09!
1A42  E5          push hl
1A43  21 E0 B4     ld hl,B4E0       KM KEYBOARD 'put back' character
1A46  7E           ld a,(hl)
1A47  36 FF        ld (hl),FF       mark empty
1A49  BE           cp (hl)          was it empty before?
1A4A  38 27        jr c,1A73        return with key in <a>
1A4C  2A DE B4     ld hl,(B4DE)     KM expansion string flag and count
1A4F  7C           ld a,h
1A50  B7           or a
1A51  20 11        jr nz,1A64       get char from expansion string
```

```
1A53  CD 5C 1B      call 1B5C        KM READ a KEY
1A56  30 1B         jr nc,1A73       no key available
1A58  FE 80         cp 80            is it an expansion token?
1A5A  38 17         jr c,1A73        return with key in <a>
1A5C  FE A0         cp A0            =160.
1A5E  3F            ccf
1A5F  38 12         jr c,1A73        return with key in <a>
1A61  67            ld h,a
1A62  2E 00         ld 1,00          <hl> builds the index
1A64  D5            push de
1A65  CD 2E 1B      call 1B2E        KM GET EXPANSION string, <a>=exp. token, <1>
1A68  38 02         jr c,1A6C        char found
1A6A  26 00         ld h,00          =0.
1A6C  2C            inc 1
1A6D  22 DE B4      ld (B4DE),hl     KM expansion string flag and count
1A70  D1            pop de
1A71  30 E0         jr nc,1A53       check keyboard again
1A73  E1            pop hl
1A74  C9            ret

----- set KEYBOARD 'put back' to be empty
      @ 1A21!
1A75  3E FF         ld a,FF          'IGNORE

----- KM RETURN CHAR <a> to 'put back' location
      @ BB0C!
1A77  32 E0 B4      ld (B4E0),a      KM KEYBOARD 'put back' character
1A7A  C9            ret

----- KM allocate EXP BUFFER (de), <hl>=len
      @ BB15!
1A7B  CD 81 1A      call 1A81        performs ALLOCATE EXPANSION BUFFER
1A7E  3F            ccf
1A7F  FB            ei
1A80  C9            ret

----- performs ALLOCATE EXPANSION BUFFER
      @ 1A2A! 1A7B!
1A81  F3            di
1A82  7D            ld a,1
1A83  D6 31         sub 31           =49.
1A85  7C            ld a,h
1A86  DE 00         sbc a,00         =0.
1A88  D8            ret c            no room
1A89  19            add hl,de
1A8A  22 E3 B4      ld (B4E3),hl     KM pointer to end of expansion buffer +1
1A8D  EB            ex de,hl
1A8E  22 E1 B4      ld (B4E1),hl     KM pointer to FUNCTION KEY EXPANSION BUFFER
1A91  01 30 0A      ld bc,0A30       b= count 10., c= '0
1A94  36 01         ld (hl),01       len of expansion string
1A96  23            inc hl
1A97  71            ld (hl),c        ='0 '1 '2 '3...
1A98  23            inc hl
1A99  0C            inc c
1A9A  10 F8         djnz 1A94        next entry
1A9C  EB            ex de,hl
1A9D  21 B3 1A      ld hl,1AB3       data for '.' and 'ENTER
1AA0  0E 0A         ld c,0A          count = 10.
1AA2  ED B0         ldir
1AA4  EB            ex de,hl
1AA5  06 13         ld b,13          count = 19.
1AA7  AF            xor a
1AA8  77            ld (hl),a        clear rest of buffer
1AA9  23            inc hl
1AAA  10 FC         djnz 1AA8        next
```

```
1AAC  22 E5 B4    ld (B4E5),hl      KM expansion buffer pointer
1AAF  32 DF B4    ld (B4DF),a       KM expansion buffer flag
1AB2  C9          ret

----- data for '.' and 'ENTER
1AB3  01 2E 01 0D 05 52 55 4E  22 0D                      '.....RUN".

----- KM SET EXPANSION string
      @ BBOF!
1ABD  F3          di
1ABE  78          ld a,b            b = expansion token
1ABF  CD 3E 1B    call 1B3E         get len and addr of expansion string
1AC2  30 1F       jr nc,1AE3        string too long
1AC4  C5          push bc
1AC5  D5           push de
1AC6  E5            push hl
1AC7  CD E5 1A        call 1AE5     shift rest of buffer up/down matching new le
1ACA  3F              ccf
1ACB  E1            pop hl
1ACC  D1           pop de
1ACD  C1          pop bc
1ACE  30 13       jr nc,1AE3        invalid token
1AD0  1B          dec de
1AD1  79          ld a,c            length of string
1AD2  0C          inc c
1AD3  12          ld (de),a         insert byte
1AD4  13          inc de
1AD5  E7          rst 4             ld a,(hl), ROMs disabled
1AD6  23          inc hl
1AD7  0D          dec c
1AD8  20 F9       jr nz,1AD3        next byte
1ADA  21 DF B4    ld hl,B4DF        KM expansion buffer flag
1ADD  78          ld a,b
1ADE  AE          xor (hl)
1ADF  20 01       jr nz,1AE2        = jump to function key routine
1AE1  77          ld (hl),a
1AE2  37          scf
1AE3  FB          ei
1AE4  C9          ret

----- shift rest of buffer up/down matching new len
1AE5  06 00       ld b,00           =0
1AE7  60          ld h,b            =0
1AE8  6F          ld l,a            = len of old entry
1AE9  79          ld a,c            = len of new entry
1AEA  95          sub l
1AEB  C8          ret z             same length, do nothing
1AEC  30 0F       jr nc,1AFD        new len is longer
1AEE  7D          ld a,l
1AEF  69          ld l,c
1AF0  4F          ld c,a
1AF1  19          add hl,de
1AF2  EB          ex de,hl
1AF3  09          add hl,bc
1AF4  CD 22 1B    call 1B22         <bc>=<expans buff ptr>-<hl>
1AF7  28 23       jr z,1B1C         ...
1AF9  ED B0       ldir
1AFB  18 1F       jr 1B1C           ...

1AFD  4F          ld c,a
1AFE  19          add hl,de
1AFF  E5          push hl
1B00  2A E5 B4     ld hl,(B4E5)     KM expansion buffer pointer
1B03  09           add hl,bc
1B04  EB           ex de,hl
```

```
1B05  2A E3 B4      ld hl,(B4E3)      KM pointer to end of expansion buffer +1
1B08  7D            ld a,l
1B09  93            sub e
1B0A  7C            ld a,h
1B0B  9A            sbc a,d
1B0C  E1           pop hl
1B0D  D8           ret c
1B0E  CD 22 1B     call 1B22          <bc>=<expans buff ptr>-<hl>
1B11  2A E5 B4     ld hl,(B4E5)       KM expansion buffer pointer
1B14  28 06        jr z,1B1C          ...
1B16  D5           push de
1B17  1B            dec de
1B18  2B            dec hl
1B19  ED B8         lddr
1B1B  D1           pop de
1B1C  ED 53 E5 B4  ld (B4E5),de       KM expansion buffer pointer
1B20  B7           or a
1B21  C9           ret

----- <bc>=<expans buff ptr>-<hl>
      @ 1AF4! 1B0E!
1B22  3A E5 B4     ld a,(B4E5)        KM expansion buffer pointer
1B25  95           sub l
1B26  4F           ld c,a
1B27  3A E6 B4     ld a,(B4E6)        expansion buffer pointer hi byte
1B2A  9C           sbc a,h
1B2B  47           ld b,a
1B2C  B1           or c
1B2D  C9           ret

----- KM GET EXPANSION string, <a>=exp. token, <l>=char#, =<a>char, =carry
      @ 1A65! BB12!
1B2E  CD 3E 1B     call 1B3E          get len and addr of expansion string
1B31  D0           ret nc
1B32  BD           cp l               char# (0...n)
1B33  C8           ret z              last char
1B34  3F           ccf
1B35  D0           ret nc
1B36  E5           push hl
1B37  26 00         ld h,00           =0.
1B39  19            add hl,de
1B3A  7E            ld a,(hl)
1B3B  E1           pop hl
1B3C  37           scf
1B3D  C9           ret

----- get len and addr of expansion string
      @ 1ABF! 1B2E!
1B3E  E6 7F        and 7F             mask out bit 7
1B40  FE 20        cp 20              max len of entry = 32.
1B42  D0           ret nc             illegal len
1B43  E5           push hl
1B44  2A E1 B4      ld hl,(B4E1)      KM pointer to FUNCTION KEY EXPANSION BUFFER
1B47  11 00 00      ld de,0000
1B4A  3C            inc a
1B4B  19            add hl,de
1B4C  5E            ld e,(hl)         get len of this entry
1B4D  23            inc hl
1B4E  3D            dec a             count
1B4F  20 FA         jr nz,1B4B        skip entry by adding its len
1B51  7B            ld a,e            len of expansion token to be replaced
1B52  EB            ex de,hl
1B53  E1           pop hl
1B54  37           scf
1B55  C9           ret
```

```
----- KM WAIT for KEY
      @ 1B59' 2771! BB18!
1B56  CD 5C 1B     call 1B5C        KM READ a KEY
1B59  30 FB        jr nc,1B56       KM WAIT for KEY
1B5B  C9           ret

----- KM READ a KEY
      @ 1A53! 1B56! BB1B!
1B5C  E5           push hl
1B5D  C5            push bc
1B5E  CD 15 1D       call 1D15      ...
1B61  30 3A          jr nc,1B9D     ...
1B63  79             ld a,c
1B64  FE EF          cp EF          'BREAK mark
1B66  28 34          jr z,1B9C      yes, return
1B68  E6 0F          and 0F         mask out
1B6A  87             add a,a
1B6B  87             add a,a
1B6C  87             add a,a        *8
1B6D  3D             dec a
1B6E  3C             inc a
1B6F  CB 08          rrc b
1B71  30 FB          jr nc,1B6E     bit not set
1B73  CD A0 1B       call 1BA0      get control, shift or translate entry
1B76  21 E8 B4       ld hl,B4E8     KM shift lock state
1B79  CB 7E          bit 7,(hl)
1B7B  28 0A          jr z,1B87      caps not locked
1B7D  FE 61          cp 61          'a
1B7F  38 06          jr c,1B87      ...
1B81  FE 7B          cp 7B          '{
1B83  30 02          jr nc,1B87     ...
1B85  C6 E0          add a,E0       change to upper case
1B87  FE FF          cp FF          'IGNORE
1B89  28 D3          jr z,1B5E      ok, get next key matrix
1B8B  FE FE          cp FE          'SHIFT LOCK
1B8D  21 E7 B4       ld hl,B4E7     KM caps lock state
1B90  28 05          jr z,1B97      yes, flip caps lock state flag
1B92  FE FD          cp FD          'CAPS LOCK
1B94  23             inc hl         shift lock state
1B95  20 05          jr nz,1B9C     no, not shift lock
1B97  7E             ld a,(hl)
1B98  2F             cpl
1B99  77             ld (hl),a
1B9A  18 C2          jr 1B5E        look for another key

1B9C  37             scf
1B9D  C1            pop bc
1B9E  E1           pop hl
1B9F  C9           ret

----- get control, shift or translate entry
      @ 1B73!
1BA0  CB 11        rl c
1BA2  DA 48 1D     jp c,1D48        KM GET CONTROL entry, in: <a>=key#, out: <a>
1BA5  47           ld b,a
1BA6  3A E7 B4     ld a,(B4E7)      KM caps lock state
1BA9  B1           or c
1BAA  E6 40        and 40           test bit 6, shift key
1BAC  78           ld a,b
1BAD  C2 43 1D     jp nz,1D43       KM GET SHIFT entry, in: <a>=key#, out: <a>=t
1BB0  C3 3E 1D     jp 1D3E          KM GET TRANSLATE, in: <a>=key#, out: <a>=tra
```

```
----- KM GET STATE <h>=caps, <l>=shift lock
      @ BB21!
1BB3  2A E7 B4    ld hl,(B4E7)      KM caps lock state
1BB6  C9          ret

----- KM update key state map (every 1/50 second)
      @ 00D9!
1BB7  11 FF B4    ld de,B4FF        KM KEY last cycle state map
1BBA  21 F5 B4    ld hl,B4F5        KM KEY change state map
1BBD  CD 46 08    call 0846         ask keys pressed and set map
1BC0  3A 01 B5    ld a,(B501)       = shift and caps bits of last scann
1BC3  E6 A0       and A0            mask 10100000
1BC5  4F          ld c,a
1BC6  21 ED B4    ld hl,B4ED        this byte contains the 'CTL and 'SHFT bits
1BC9  B6          or (hl)
1BCA  77          ld (hl),a
1BCB  21 FF B4    ld hl,B4FF        KM KEY last cycle state map
1BCE  11 EB B4    ld de,B4EB        KM key state map (marks pressed keys by sett
1BD1  06 00       ld b,00           =0.
1BD3  1A          ld a,(de)
1BD4  AE          xor (hl)
1BD5  A6          and (hl)
1BD6  C4 48 1C    call nz,1C48      ...
1BD9  7E          ld a,(hl)
1BDA  12          ld (de),a
1BDB  23          inc hl
1BDC  13          inc de
1BDD  0C          inc c
1BDE  79          ld a,c
1BDF  E6 0F       and 0F            =15.
1BE1  FE 0A       cp 0A             =10.
1BE3  20 EE       jr nz,1BD3        ...
1BE5  79          ld a,c
1BE6  E6 A0       and A0            =160.
1BE8  CB 71       bit 6,c
1BEA  4F          ld c,a
1BEB  C4 EE BD    call nz,BDEE      KM TEST BREAK or reset; in: interrupts disab
1BEE  78          ld a,b
1BEF  B7          or a
1BF0  C0          ret nz
1BF1  21 09 B5    ld hl,B509        KM time count for repeat speed
1BF4  35          dec (hl)
1BF5  C0          ret nz
1BF6  2A 0A B5    ld hl,(B50A)      ...
1BF9  EB          ex de,hl
1BFA  42          ld b,d
1BFB  16 00       ld d,00           =0.
1BFD  21 EB B4    ld hl,B4EB        KM key state map (marks pressed keys by sett
1C00  19          add hl,de
1C01  7E          ld a,(hl)
1C02  2A 47 B5    ld hl,(B547)      KM repeat key, pointer to table
1C05  19          add hl,de
1C06  A6          and (hl)
1C07  A0          and b
1C08  C8          ret z             this key may not repeat
1C09  21 09 B5    ld hl,B509        KM time count for repeat speed
1C0C  34          inc (hl)
1C0D  3A 40 B5    ld a,(B540)       ...
1C10  B7          or a
1C11  C0          ret nz
1C12  79          ld a,c
1C13  B3          or e
1C14  4F          ld c,a
1C15  3A E9 B4    ld a,(B4E9)       KM KEY repeat speed
```

```
      @ 1C52!
1C18  32 09 B5    ld (B509),a       KM time count for repeat speed
1C1B  CD FE 1C    call 1CFE         ...
1C1E  79          ld a,c
1C1F  E6 0F       and 0F            =15.
1C21  6F          ld l,a
1C22  60          ld h,b
1C23  22 0A B5    ld (B50A),hl      ...
1C26  FE 08       cp 08             =8.
1C28  C0          ret nz
1C29  CB 60       bit 4,b
1C2B  C0          ret nz
1C2C  CB F1       set 6,c
1C2E  C9          ret

----- KM TEST BREAK or reset; in: interrupts disabled, <c>=shft/ctl key states
      @ 1A39  BDEE
1C2F  21 F3 B4    ld hl,B4F3        bit 2 = break key
1C32  CB 56       bit 2,(hl)
1C34  C8          ret z
1C35  79          ld a,c
1C36  EE A0       xor A0            test for 'SHIFT and 'CTL keys
1C38  20 56       jr nz,1C90        KM BREAK EVENT
1C3A  C5          push bc
1C3B  23           inc hl
1C3C  06 0A        ld b,0A          =10.
1C3E  8E           adc a,(hl)
1C3F  2B           dec hl
1C40  10 FC        djnz 1C3E        ...
1C42  C1          pop bc
1C43  FE A4       cp A4             =164.
1C45  20 49       jr nz,1C90        KM BREAK EVENT
1C47  C7          rst 0

      @ 1BD6!
1C48  E5          push hl
1C49  D5           push de
1C4A  5F            ld e,a
1C4B  2F            cpl
1C4C  3C            inc a
1C4D  A3            and e
1C4E  47            ld b,a
1C4F  3A EA B4      ld a,(B4EA)     KM KEY startup delay
1C52  CD 18 1C      call 1C18       ...
1C55  78            ld a,b
1C56  AB            xor e
1C57  20 F1         jr nz,1C4A      ...
1C59  D1           pop de
1C5A  E1          pop hl
1C5B  C9          ret

----- KM GET JOYSTICKs 1=<h>, 2=<l>
      @ BB24!
1C5C  3A F1 B4    ld a,(B4F1)       joystick 2
1C5F  E6 7F       and 7F            mask out bit 7
1C61  6F          ld l,a
1C62  3A F4 B4    ld a,(B4F4)       joystick 1
1C65  E6 7F       and 7F            mask out bit 7
1C67  67          ld h,a
1C68  C9          ret
```

```
----- KM GET DELAY key, <h>=start, <l>=rep. speed
      @ BB42!
1C69  2A E9 B4     ld hl,(B4E9)     KM KEY repeat speed
1C6C  C9           ret

----- KM SET DELAY key, <h>=start, <l>=rep. speed
      @ 19E3! BB3F!
1C6D  22 E9 B4     ld (B4E9),hl     KM KEY repeat speed
1C70  C9           ret

----- KM ARM BREAK, (de)=routine, <c>=ROM select
      @ BB45!
1C71  CD 82 1C     call 1C82        KM DISARM BREAK
1C74  21 0D B5     ld hl,B50D       KM event block BREAK
1C77  06 40        ld b,40          event class
1C79  CD D2 01     call 01D2        KL INIT EVENT BLOCK (hl)=block, <b>=class, <
1C7C  3E FF        ld a,FF          set
1C7E  32 0C B5     ld (B50C),a      KM BREAK ENABLE FLAG
1C81  C9           ret

----- KM DISARM BREAK
      @ 1A33  1C71! BB48!
1C82  C5           push bc
1C83  D5            push de
1C84  21 0C B5       ld hl,B50C     KM BREAK ENABLE FLAG
1C87  36 00          ld (hl),00     reset BREAK flag
1C89  23             inc hl
1C8A  CD 85 02       call 0285      KL DEL SYNC, delete block (hl) from queue
1C8D  D1            pop de
1C8E  C1           pop bc
1C8F  C9           ret

----- KM BREAK EVENT
      @ 1C38' 1C45' BB4B!
1C90  21 0C B5     ld hl,B50C       KM BREAK ENABLE FLAG
1C93  7E           ld a,(hl)
1C94  36 00        ld (hl),00       reset BREAK ENABLE FLAG
1C96  BE           cp (hl)          was it set before?
1C97  C8           ret z            no, it was reset; return
1C98  C5           push bc
1C99  D5            push de
1C9A  23             inc hl
1C9B  CD E2 01       call 01E2      KL EVENT, kick an event block (hl)
1C9E  0E EF          ld c,EF        'BREAK mark
1CA0  CD FE 1C       call 1CFE      ...
1CA3  D1            pop de
1CA4  C1           pop bc
1CA5  C9           ret

----- KM GET REPEAT key# <a>, nz if repeat
      @ BB3C!
1CA6  2A 47 B5     ld hl,(B547)     KM repeat key, pointer to table
1CA9  18 1D        jr 1CC8

----- KM SET REPEAT key# <a>, <b>=0 = not
      @ BB39!
1CAB  FE 50        cp 50            max key number = 80.
1CAD  D0           ret nc
1CAE  2A 47 B5     ld hl,(B547)     KM repeat key, pointer to table
1CB1  CD CD 1C     call 1CCD        set bit in <a> to mask or compare
1CB4  4F           ld c,a
1CB5  2F           cpl
1CB6  A6           and (hl)
1CB7  77           ld (hl),a
1CB8  79           ld a,c
```

```
1CB9  A0          and b
1CBA  B6          or (hl)
1CBB  77          ld (hl),a
1CBC  C9          ret

----- KM TEST if KEY #<a> is pressed
      @ 2A79! BB1E!
1CBD  F5          push af
1CBE  3A ED B4     ld a,(B4ED)      this byte contains the 'CTL and 'SHFT bits
1CC1  E6 A0        and A0           10100000 mask 'CTL (1B7) and 'SHFT (1B5)
1CC3  4F           ld c,a
1CC4  F1          pop af
1CC5  21 EB B4    ld hl,B4EB        KM key state map (marks pressed keys by sett
1CC8  CD CD 1C    call 1CCD         set bit in <a> to mask or compare
1CCB  A6          and (hl)
1CCC  C9          ret

----- set bit in <a> to mask or compare
      @ 1CB1! 1CC8!
1CCD  D5          push de
1CCE  F5           push af
1CCF  E6 F8         and F8          preserve upper bits to calc addr
1CD1  0F            rrca
1CD2  0F            rrca
1CD3  0F            rrca
1CD4  5F            ld e,a
1CD5  16 00         ld d,00
1CD7  19            add hl,de
1CD8  F1           pop af
1CD9  E5           push hl
1CDA  21 E5 1C      ld hl,1CE5      bit map masks
1CDD  E6 07         and 07          preserve lower bits
1CDF  5F            ld e,a
1CE0  19            add hl,de
1CE1  7E            ld a,(hl)
1CE2  E1           pop hl
1CE3  D1          pop de
1CE4  C9          ret

----- bit map masks
      @ 1CDA:
1CE5  01 02 04 08 10 20 40 80

----- performs reset KM, part 1
      @ 1A1E!
1CED  F3          di
1CEE  21 3C B5    ld hl,B53C        ...
1CF1  36 15       ld (hl),15
1CF3  23          inc hl
1CF4  AF          xor a
1CF5  77          ld (hl),a
1CF6  23          inc hl
1CF7  36 01       ld (hl),01
1CF9  23          inc hl
1CFA  77          ld (hl),a
1CFB  23          inc hl
1CFC  77          ld (hl),a
1CFD  C9          ret

      @ 1C1B! 1CA0!
1CFE  21 3C B5    ld hl,B53C        ...
1D01  B7          or a
1D02  35          dec (hl)
1D03  28 0E       jr z,1D13         ...
1D05  CD 2C 1D    call 1D2C         ...
```

```
1D08  71          ld (hl),c
1D09  23          inc hl
1D0A  70          ld (hl),b
1D0B  21 40 B5    ld hl,B540        ...
1D0E  34          inc (hl)
1D0F  21 3E B5    ld hl,B53E        ...
1D12  37          scf
1D13  34          inc (hl)
1D14  C9          ret

      @ 1B5E!
1D15  21 3E B5    ld hl,B53E        ...
1D18  B7          or a
1D19  35          dec (hl)
1D1A  28 0E       jr z,1D2A         restore
1D1C  CD 2C 1D    call 1D2C         ...
1D1F  4E          ld c,(hl)
1D20  23          inc hl
1D21  46          ld b,(hl)
1D22  21 40 B5    ld hl,B540        ...
1D25  35          dec (hl)
1D26  21 3C B5    ld hl,B53C        ...
1D29  37          scf
1D2A  34          inc (hl)
1D2B  C9          ret

      @ 1D05! 1D1C!
1D2C  23          inc hl
1D2D  34          inc (hl)
1D2E  7E          ld a,(hl)
1D2F  FE 14       cp 14             =20.
1D31  20 02       jr nz,1D35        ...
1D33  AF          xor a
1D34  77          ld (hl),a
1D35  87          add a,a
1D36  CE 14       adc a,14
1D38  6F          ld l,a
1D39  CE B5       adc a,B5          <hl>=<a>*2+B514
1D3B  95          sub l
1D3C  67          ld h,a
1D3D  C9          ret

----- KM GET TRANSLATE, in: <a>=key#, out: <a>=translation  .
      @ 1BB0  BB2A!
1D3E  2A 41 B5    ld hl,(B541)      KM translate normal entry, pointer
1D41  18 08       jr 1D4B

----- KM GET SHIFT entry, in: <a>=key#, out: <a>=translation
      @ 1BAD  BB30!
1D43  2A 43 B5    ld hl,(B543)      KM translate shift entry, pointer
1D46  18 03       jr 1D4B

----- KM GET CONTROL entry, in: <a>=key#, out: <a>=translation
      @ 1BA2  BB36!
1D48  2A 45 B5    ld hl,(B545)      KM translate control entry, pointer
1D4B  85          add a,l           add key# to start of table
1D4C  6F          ld l,a
1D4D  8C          adc a,h
1D4E  95          sub l
1D4F  67          ld h,a
1D50  7E          ld a,(hl)         get translation
1D51  C9          ret
```

```
----- KM SET TRANSLATE entry, <a>=key#, <b>=new translation
      @ BB27!
1D52  2A 41 B5      ld hl,(B541)      KM translate normal entry, pointer
1D55  18 08         jr 1D5F

----- KM SET SHIFT entry, <a>=key#, <b>=new translation
      @ BB2D!
1D57  2A 43 B5      ld hl,(B543)      KM translate shift entry, pointer
1D5A  18 03         jr 1D5F

----- KM SET CONTROL entry, <a>=key#, <b>=new translation
      @ BB33!
1D5C  2A 45 B5      ld hl,(B545)      KM translate control entry, pointer
1D5F  FE 50         cp 50             max key# =79.
1D61  D0            ret nc
1D62  85            add a,l           add key# to start of table
1D63  6F            ld l,a
1D64  8C            adc a,h
1D65  95            sub l
1D66  67            ld h,a
1D67  70            ld (hl),b         set entry to <b>
1D68  C9            ret

----- default KEY normal/shift/control/repeat entries (copied to B34C ... B445)
      @ 1A05:
1D69  F0 F3 F1 89 86 83 8B 8A  F2 E0 87 88 85 81 82 80  10 5B 0D 5D 84 FF 5C FF
1D81  5E 2D 40 70 3B 3A 2F 2E  30 39 6F 69 6C 6B 6D 2C  38 37 75 79 68 6A 6E 20
1D99  36 35 72 74 67 66 62 76  34 33 65 77 73 64 63 78  31 32 FC 71 09 61 FD 7A
1DB1  0B 0A 08 09 58 5A FF 7F  F4 F7 F5 89 86 83 8B 8A  F6 E0 87 88 85 81 82 80
1DC9  10 7B 0D 7D 84 FF 60 FF  A3 3D 7C 50 2B 2A 3F 3E  5F 29 4F 49 4C 4B 4D 3C
1DE1  28 27 55 59 48 4A 4E 20  26 25 52 54 47 46 42 56  24 23 45 57 53 44 43 58
1DF9  21 22 FC 51 09 41 FD 5A  0B 0A 08 09 58 5A FF 7F  F8 FB F9 89 86 83 8C 8A
1E11  FA E0 87 88 85 81 82 80  10 1B 0D 1D 84 FF 1C FF  1E FF 00 10 FF FF FF FF
1E29  1F FF 0F 09 0C 0B 0D FF  FF FF 15 19 08 0A 0E FF  FF FF 12 14 07 06 02 16
1E41  FF FF 05 17 13 04 03 18  FF 7E FC 11 E1 01 FE 1A  FF FF FF FF FF FF FF 7F
1E59  07 03 4B FF FF FF FF FF  AB 8F

-----
1E63  C7 C7 C7 C7 C7
```

```
----- SOUND RESET
      @ 05E0! 0640! 287A! BCA7!
1E68  AF            xor a
1E69  F3            di
1E6A  32 52 B5      ld (B552),a       SOUND channel bits of active sounds
1E6D  32 51 B5      ld (B551),a       SOUND save for active sounds
1E70  21 55 B5      ld hl,B555        SOUND event block
1E73  11 03 1F      ld de,1F03        event routine SOUND
1E76  06 81         ld b,81           =129.
1E78  CD D2 01      call 01D2         KL INIT EVENT BLOCK (hl)=block, <b>=class, <
1E7B  3E 3F         ld a,3F           =63.
1E7D  32 19 B6      ld (B619),a       ...
1E80  21 5C B5      ld hl,B55C        SOUND QUEUE, channel A, (first entry), chann
1E83  01 3D 00      ld bc,003D        ...
1E86  11 08 01      ld de,0108        ...
1E89  AF            xor a
1E8A  77            ld (hl),a
1E8B  23            inc hl
1E8C  72            ld (hl),d
1E8D  23            inc hl
1E8E  73            ld (hl),e
1E8F  09            add hl,bc
1E90  3C            inc a
1E91  EB            ex de,hl
1E92  29            add hl,hl
1E93  EB            ex de,hl
1E94  FE 03         cp 03             =3.
1E96  20 F2         jr nz,1E8A        ...
1E98  0E 07         ld c,07           =7.

----- flush sound queues of channel(s) <c>
      @ 1FA9!
1E9A  DD E5         push ix
1E9C  E5             push hl
1E9D  21 1D B5        ld hl,B51D      (+3F=B55C)=start of sound queue chan A
1EA0  41              ld b,c
1EA1  11 3F 00        ld de,003F      len of one channel block
1EA4  19              add hl,de
1EA5  CB 38           srl b
1EA7  30 F8           jr nc,1EA1      next channel
1EA9  C5              push bc
1EAA  E5               push hl
1EAB  DD E1            pop ix
1EAD  EB               ex de,hl
1EAE  CD 7F 22         call 227F      ...
1EB1  13               inc de
1EB2  13               inc de
1EB3  13               inc de
1EB4  6B               ld l,e         hl=de
1EB5  62               ld h,d
1EB6  13               inc de
1EB7  01 3B 00         ld bc,003B
1EBA  36 00            ld (hl),00     =0.
1EBC  ED B0            ldir
1EBE  DD 36 1C 04      ld (ix+1C),04 =4.
1EC2  C1              pop bc
1EC3  EB              ex de,hl
1EC4  04              inc b
1EC5  10 DE           djnz 1EA5       next entry in queue
1EC7  E1             pop hl
1EC8  DD E1         pop ix
1ECA  C9            ret
```

```
----- SOUND HOLD, stop all sounds
      @ BCB6!
1ECB  21 52 B5     ld hl,B552        SOUND channel bits of active sounds
1ECE  F3           di
1ECF  7E           ld a,(hl)
1ED0  36 00        ld (hl),00        mark sounds not active
1ED2  FB           ei
1ED3  B7           or a
1ED4  C8           ret z             not active before, return
1ED5  2B           dec hl
1ED6  77           ld (hl),a         save previously active sound bits
1ED7  2E 03        ld l,03           max # of active sounds
1ED9  0E 00        ld c,00
1EDB  3E 07        ld a,07
1EDD  85           add a,l           <a>= A, 9, 8
1EDE  CD 26 08     call 0826         MC SOUND REGISTER, send <a>=reg#, <c>=data
1EE1  2D           dec l
1EE2  20 F7        jr nz,1EDB        next
1EE4  37           scf
1EE5  C9           ret

----- SOUND CONTINUE stopped sounds
      @ 1F9F! 204B! BCB9!
1EE6  3A 51 B5     ld a,(B551)       SOUND save for active sounds
1EE9  B7           or a
1EEA  C8           ret z             nothing to restore
1EEB  DD 21 1D B5 ld ix,B51D         (+3F=B55C)=start of sound queue chan A
1EEF  11 3F 00     ld de,003F        len of one channel block
1EF2  DD 19        add ix,de
1EF4  CB 3F        srl a
1EF6  F5           push af
1EF7  DD 7E 0F      ld a,(ix+0F)     get sound for this channel
1EFA  DC 76 22      call c,2276      ...
1EFD  F1           pop af
1EFE  20 F2        jr nz,1EF2        restore next channel
1F00  C3 1E 20     jp 201E           ...

----- event routine SOUND
      @ 1E73: B559:
1F03  DD E5        push ix
1F05  21 50 B5      ld hl,B550       SOUND flag ??
1F08  E5             push hl
1F09  AF              xor a
1F0A  77              ld (hl),a
1F0B  23              inc hl
1F0C  46              ld b,(hl)      save for active sounds
1F0D  C5              push bc
1F0E  23               inc hl
1F0F  B6               or (hl)       channel bits of active sounds
1F10  28 22            jr z,1F34     ...
1F12  DD 21 1D B5      ld ix,B51D    (+3F=B55C)=start of sound queue chan A
1F16  01 3F 00         ld bc,003F    len of one channel block
1F19  DD 09            add ix,bc
1F1B  CB 3F            srl a
1F1D  30 FA            jr nc,1F19    ...
1F1F  F5               push af
1F20  DD 7E 04          ld a,(ix+04)
1F23  1F                rra
1F24  DC C2 22          call c,22C2  ...
1F27  DD 7E 07          ld a,(ix+07)
1F2A  1F                rra
1F2B  DC B6 21          call c,21B6  ...
1F2E  DC A8 20          call c,20A8  ...
1F31  F1               pop af
1F32  20 E2            jr nz,1F16    another channel
```

```
1F34  C1           pop bc
1F35  E1           pop hl
1F36  7E           ld a,(hl)
1F37  B7           or a
1F38  28 20        jr z,1F5A         ...
1F3A  4F           ld c,a
1F3B  23           inc hl
1F3C  7E           ld a,(hl)
1F3D  70           ld (hl),b
1F3E  A8           xor b
1F3F  47           ld b,a
1F40  23           inc hl
1F41  B6           or (hl)
1F42  77           ld (hl),a
1F43  78           ld a,b
1F44  2F           cpl
1F45  A1           and c
1F46  28 12        jr z,1F5A         ...
1F48  DD 21 1D B5  ld ix,B51D        (+3F=B55C)=start of sound queue chan A
1F4C  11 3F 00     ld de,003F        len of one channel block
1F4F  DD 19        add ix,de
1F51  CB 3F        srl a
1F53  F5           push af
1F54  DC 7F 22     call c,227F       ...
1F57  F1           pop af
1F58  20 F5        jr nz,1F4F        ...
1F5A  AF           xor a
1F5B  32 54 B5     ld (B554),a       SOUND rendenzvous byte ??
1F5E  DD E1        pop ix
1F60  C9           ret

----- SOUND TICK (every 1/300 second)
      @ 00CF!
1F61  21 52 B5     ld hl,B552        SOUND channel bits of active sounds
1F64  7E           ld a,(hl)
1F65  B7           or a              any sounds active?
1F66  C8           ret z             no, return
1F67  23           inc hl
1F68  35           dec (hl)          decrement SOUND TIMER count
1F69  C0           ret nz            countdown not finished; return
1F6A  34           inc (hl)          set SOUND TIMER to one
1F6B  23           inc hl
1F6C  7E           ld a,(hl)
1F6D  B7           or a
1F6E  C0           ret nz
1F6F  2B           dec hl
1F70  36 03        ld (hl),03        recharge value for SOUND TIMER
1F72  2B           dec hl
1F73  46           ld b,(hl)         get bits of active sounds
1F74  21 22 B5     ld hl,B522        (+3F=B561)=noise period chan A
1F77  11 3F 00     ld de,003F        len of one channel block
1F7A  AF           xor a             =0
1F7B  19           add hl,de
1F7C  CB 38        srl b
1F7E  30 FB        jr nc,1F7B        find an active sound channel
1F80  35           dec (hl)
1F81  20 05        jr nz,1F88        ...
1F83  2B           dec hl
1F84  CB 06        rlc (hl)
1F86  8A           adc a,d
1F87  23           inc hl
1F88  23           inc hl
1F89  35           dec (hl)          decrement sound timer again
1F8A  20 05        jr nz,1F91        ...
1F8C  23           inc hl
```

```
1F8D  CB 06       rlc (hl)
1F8F  8A          adc a,d
1F90  2B          dec hl
1F91  2B          dec hl
1F92  04          inc b
1F93  10 E6       djnz 1F7B          index up to next sound block
1F95  B7          or a
1F96  C8          ret z
1F97  21 54 B5    ld hl,B554         SOUND rendenzvous byte ??
1F9A  77          ld (hl),a
1F9B  23          inc hl
1F9C  C3 E2 01    jp 01E2            KL EVENT, kick an event block (hl)

----- SOUND QUEUE, add a sound, (hl)=sound program
      @ 14DD! BCAA!
1F9F  CD E6 1E    call 1EE6          SOUND CONTINUE stopped sounds
1FA2  7E          ld a,(hl)
1FA3  E6 07       and 07             mask out all but channels
1FA5  37          scf
1FA6  C8          ret z              no channels specified
1FA7  4F          ld c,a
1FA8  B6          or (hl)
1FA9  FC 9A 1E    call m,1E9A        flush sound queues of channel(s) <c>
1FAC  41          ld b,c
1FAD  DD 21 1D B5 ld ix,B51D         (+3F=B55C)=start of sound queue chan A
1FB1  11 3F 00    ld de,003F         len of one channel block
1FB4  AF          xor a
1FB5  DD 19       add ix,de
1FB7  CB 38       srl b
1FB9  30 FA       jr nc,1FB5         next channel
1FBB  DD 72 1E    ld (ix+1E),d       =0
1FBE  DD BE 1C    cp (ix+1C)
1FC1  3F          ccf
1FC2  9F          sbc a,a
1FC3  04          inc b
1FC4  10 EF       djnz 1FB5          ...
1FC6  B7          or a
1FC7  C0          ret nz
1FC8  41          ld b,c
1FC9  7E          ld a,(hl)
1FCA  1F          rra
1FCB  1F          rra
1FCC  1F          rra
1FCD  B0          or b
1FCE  E6 0F       and 0F             mask out
1FD0  4F          ld c,a
1FD1  23          inc hl
1FD2  DD 21 1D B5 ld ix,B51D         (+3F=B55C)=start of sound queue chan A
1FD6  11 3F 00    ld de,003F         len of one channel block
1FD9  DD 19       add ix,de
1FDB  CB 38       srl b
1FDD  30 FA       jr nc,1FD9         ...
1FDF  E5          push hl
1FE0  C5           push bc
1FE1  DD 7E 1B      ld a,(ix+1B)
1FE4  DD 34 1B      inc (ix+1B)
1FE7  DD 35 1C      dec (ix+1C)
1FEA  EB            ex de,hl
1FEB  CD 3A 20      call 203A        ...
1FEE  E5            push hl
1FEF  EB             ex de,hl
1FF0  DD 7E 01       ld a,(ix+01)
1FF3  2F             cpl
1FF4  A1             and c
1FF5  12             ld (de),a
```

```
1FF6  13              inc de
1FF7  7E              ld a,(hl)
1FF8  23              inc hl
1FF9  87              add a,a
1FFA  87              add a,a
1FFB  87              add a,a
1FFC  87              add a,a
1FFD  47              ld b,a
1FFE  7E              ld a,(hl)
1FFF  23              inc hl
2000  E6 0F           and 0F          mask out
2002  B0              or b
2003  12              ld (de),a
2004  13              inc de
2005  01 06 00        ld bc,0006      byte count
2008  ED B0           ldir
200A  E1             pop hl
200B  F3             di
200C  DD 7E 1A       ld a,(ix+1A)
200F  DD 34 1A       inc (ix+1A)
2012  DD B6 03       or (ix+03)
2015  FB             ei
2016  CC BD 20       call z,20BD      ...
2019  C1            pop bc
201A  E1           pop hl
201B  04           inc b              inc for following decrement
201C  10 B8        djnz 1FD6          add this sound to another queue?

      @ 1F00  206A'
201E  E5           push hl
201F  21 51 B5      ld hl,B551        SOUND save for active sounds
2022  7E            ld a,(hl)
2023  B7            or a
2024  28 11         jr z,2037         ...
2026  36 00         ld (hl),00        =0.
2028  F3            di
2029  23            inc hl
202A  46            ld b,(hl)
202B  B0            or b
202C  77            ld (hl),a
202D  78            ld a,b
202E  B7            or a
202F  20 05         jr nz,2036        ...
2031  23            inc hl
2032  36 03         ld (hl),03        =3.
2034  23            inc hl
2035  77            ld (hl),a
2036  FB            ei
2037  E1           pop hl
2038  37           scf
2039  C9           ret

      @ 1FEB! 20BA!
203A  E6 03        and 03             =3.
203C  87           add a,a
203D  87           add a,a
203E  87           add a,a
203F  C6 1F        add a,1F           =31.
2041  DD E5        push ix
2043  E1           pop hl
2044  85           add a,l
2045  6F           ld l,a
2046  8C           adc a,h
2047  95           sub l
2048  67           ld h,a
```

```
2049  C9          ret

----- SOUND RELEASE, <a>=channel(s)
      @ BCB3!
204A  6F          ld l,a
204B  CD E6 1E    call 1EE6         SOUND CONTINUE stopped sounds
204E  7D          ld a,l
204F  E6 07       and 07            =7.
2051  C8          ret z
2052  DD 21 1D B5 ld ix,B51D        (+3F=B55C)=start of sound queue chan A
2056  11 3F 00    ld de,003F        len of one channel block
2059  DD 19       add ix,de
205B  CB 3F       srl a
205D  30 FA       jr nc,2059        ...
205F  F5          push af
2060  DD CB 03 5E  bit 3,(ix+03)
2064  C4 B7 20     call nz,20B7     ...
2067  F1          pop af
2068  20 EC       jr nz,2056        ...
206A  18 B2       jr 201E           ...

----- SOUND CHECK for space in <a>, <a>=status
      @ BCAD!
206C  E6 07       and 07            mask out
206E  C8          ret z             no channels specified
206F  21 20 B5    ld hl,B520        (+3F=B55F)=tone env to use chan A
2072  11 3F 00    ld de,003F        len of one channel block
2075  19          add hl,de
2076  1F          rra
2077  30 FC       jr nc,2075        ...
2079  F3          di
207A  7E          ld a,(hl)
207B  87          add a,a
207C  87          add a,a
207D  87          add a,a
207E  11 19 00    ld de,0019        ...
2081  19          add hl,de
2082  B6          or (hl)
2083  23          inc hl
2084  23          inc hl
2085  36 00       ld (hl),00        =0.
2087  FB          ei
2088  C9          ret

----- SOUND ARM EVENT, <a>=channels, (hl)=event block
      @ BCB0!
2089  E6 07       and 07            mask out
208B  C8          ret z             no channel
208C  EB          ex de,hl
208D  21 39 B5    ld hl,B539        +3F=B578
2090  01 3F 00    ld bc,003F        len of one channel block
2093  09          add hl,bc
2094  1F          rra
2095  30 FC       jr nc,2093        next channel
2097  AF          xor a
2098  F3          di
2099  BE          cp (hl)
209A  23          inc hl
209B  73          ld (hl),e
209C  23          inc hl
209D  20 03       jr nz,20A2        ...
209F  72          ld (hl),d
20A0  FB          ei
20A1  C9          ret
```

```
20A2  77          ld (hl),a
20A3  FB          ei
20A4  EB          ex de,hl
20A5  C3 E2 01    jp 01E2          KL EVENT, kick an event block (hl)

      @ 1F2E!
20A8  DD 7E 1A    ld a,(ix+1A)
20AB  B7          or a
20AC  CA 7F 22    jp z,227F        ...
20AF  DD 7E 01    ld a,(ix+01)
20B2  21 50 B5    ld hl,B550       SOUND flag ??
20B5  B6          or (hl)
20B6  77          ld (hl),a

      @ 2064! 215B! 2165!
20B7  DD 7E 19    ld a,(ix+19)
20BA  CD 3A 20    call 203A        ...

      @ 2016!
20BD  7E          ld a,(hl)
20BE  B7          or a
20BF  28 0C       jr z,20CD        ...
20C1  CB 5F       bit 3,a
20C3  20 53       jr nz,2118       ...
20C5  E5          push hl
20C6  36 00        ld (hl),00      =0.
20C8  CD 1F 21     call 211F       ...
20CB  E1          pop hl
20CC  D0          ret nc
20CD  DD 36 03 10 ld (ix+03),10    =16.
20D1  23          inc hl
20D2  7E          ld a,(hl)
20D3  E6 F0       and F0           =240.
20D5  F5          push af
20D6  AE           xor (hl)
20D7  5F           ld e,a
20D8  23           inc hl
20D9  4E           ld c,(hl)
20DA  23           inc hl
20DB  56           ld d,(hl)
20DC  23           inc hl
20DD  B2           or d
20DE  B1           or c
20DF  28 08        jr z,20E9       ...
20E1  E5           push hl
20E2  CD AB 22      call 22AB      ...
20E5  DD 56 01      ld d,(ix+01)
20E8  E1           pop hl
20E9  4E           ld c,(hl)
20EA  23           inc hl
20EB  5E           ld e,(hl)
20EC  23           inc hl
20ED  7E           ld a,(hl)
20EE  23           inc hl
20EF  66           ld h,(hl)
20F0  6F           ld l,a
20F1  F1          pop af
20F2  CD 75 21    call 2175        ...
20F5  21 51 B5    ld hl,B551       SOUND save for active sounds
20F8  DD 7E 01    ld a,(ix+01)
20FB  B6          or (hl)
20FC  77          ld (hl),a
20FD  DD 34 19    inc (ix+19)
2100  DD 35 1A    dec (ix+1A)
2103  DD 34 1C    inc (ix+1C)
```

```
2106  F3           di
2107  DD 7E 1E     ld a,(ix+1E)
210A  DD 36 1E 00  ld (ix+1E),00     =0.
210E  FB           ei
210F  B7           or a
2110  C8           ret z
2111  67           ld h,a
2112  DD 6E 1D     ld l,(ix+1D)
2115  C3 E2 01     jp 01E2           KL EVENT, kick an event block (hl)

2118  CB 9E        res 3,(hl)
211A  DD 36 03 08  ld (ix+03),08     =8.
211E  C9           ret

      @ 20C8!
211F  DD E5        push ix
2121  47            ld b,a
2122  DD 4E 01      ld c,(ix+01)
2125  DD 21 5C B5   ld ix,B55C       SOUND QUEUE, channel A, (first entry), chann
2129  CB 47         bit 0,a
212B  20 0C         jr nz,2139       ...
212D  DD 21 9B B5   ld ix,B59B       SOUND QUEUE, channel B
2131  CB 4F         bit 1,a
2133  20 04         jr nz,2139       ...
2135  DD 21 DA B5   ld ix,B5DA       SOUND QUEUE, channel C
2139  F3            di
213A  DD 7E 03      ld a,(ix+03)
213D  A1            and c
213E  28 2D         jr z,216D        ...
2140  78            ld a,b
2141  DD BE 01      cp (ix+01)
2144  28 1A         jr z,2160        ...
2146  DD E5         push ix
2148  DD 21 DA B5    ld ix,B5DA      SOUND QUEUE, channel C
214C  CB 57          bit 2,a
214E  20 04          jr nz,2154      ...
2150  DD 21 9B B5    ld ix,B59B      SOUND QUEUE, channel B
2154  DD 7E 03       ld a,(ix+03)
2157  A1             and c
2158  28 12          jr z,216C       ...
215A  FB             ei
215B  CD B7 20       call 20B7       ...
215E  DD E1         pop ix
2160  DD 36 03 00   ld (ix+03),00    =0.
2164  FB            ei
2165  CD B7 20      call 20B7        ...
2168  DD E1        pop ix
216A  37           scf
216B  C9           ret

216C  E1            pop hl
216D  DD E1        pop ix
216F  DD 70 03     ld (ix+03),b
2172  FB           ei
2173  B7           or a
2174  C9           ret

      @ 20F2!
2175  CB FB        set 7,e
2177  DD 73 0F     ld (ix+0F),e
217A  5F           ld e,a
217B  7D           ld a,l
217C  B4           or h
217D  20 01        jr nz,2180        ...
217F  2B           dec hl
```

```
2180  DD 75 08     ld (ix+08),l
2183  DD 74 09     ld (ix+09),h
2186  79           ld a,c
2187  B7           or a
2188  28 08        jr z,2192          ...
218A  3E 06        ld a,06            =6.
218C  CD 26 08     call 0826          MC SOUND REGISTER, send <a>=reg#, <c>=data
218F  DD 7E 02     ld a,(ix+02)
2192  B2           or d
2193  CD 8B 22     call 228B          ...
2196  7B           ld a,e
2197  B7           or a
2198  28 0A        jr z,21A4          ...
219A  21 0A B6     ld hl,B60A         SOUND amplitude envelope
219D  16 00        ld d,00            =0.
219F  19           add hl,de
21A0  7E           ld a,(hl)
21A1  B7           or a
21A2  20 03        jr nz,21A7         ...
21A4  21 B2 21     ld hl,21B2         ...
21A7  DD 75 0A     ld (ix+0A),l
21AA  DD 74 0B     ld (ix+0B),h
21AD  CD 65 22     call 2265          ...
21B0  18 0D        jr 21BF            ...

      @ 21A4: 2232: 224F:
21B2  01 01 00     ld bc,0001         ...
21B5  C8           ret z

      @ 1F2B!
21B6  DD 6E 0D     ld l,(ix+0D)
21B9  DD 66 0E     ld h,(ix+0E)
21BC  DD 5E 10     ld e,(ix+10)

      @ 21B0' 2230' 2238'
21BF  7B           ld a,e
21C0  FE FF        cp FF              =255.
21C2  28 76        jr z,223A          ...
21C4  87           add a,a
21C5  7E           ld a,(hl)
21C6  23           inc hl
21C7  38 4A        jr c,2213          ...
21C9  28 0D        jr z,21D8          ...
21CB  1D           dec e
21CC  B7           or a
21CD  20 06        jr nz,21D5         ...
21CF  DD B6 0F     or (ix+0F)
21D2  F2 DD 21     jp p,21DD          ...
21D5  DD 86 0F     add a,(ix+0F)
21D8  E6 0F        and 0F             =15.
21DA  CD 73 22     call 2273          ...

      @ 21D2
21DD  4E           ld c,(hl)
21DE  DD 7E 09     ld a,(ix+09)
21E1  47           ld b,a
21E2  87           add a,a
21E3  38 1B        jr c,2200          ...
21E5  AF           xor a
21E6  91           sub c
21E7  DD 86 08     add a,(ix+08)
21EA  38 0C        jr c,21F8          ...
21EC  05           dec b
21ED  F2 F5 21     jp p,21F5          ...
21F0  DD 4E 08     ld c,(ix+08)
```

```
21F3  AF          xor a
21F4  47          ld b,a

      @ 21ED
21F5  DD 70 09    ld (ix+09),b
21F8  DD 77 08    ld (ix+08),a
21FB  B0          or b
21FC  20 02       jr nz,2200        ...
21FE  1E FF       ld e,FF           =255.
2200  7B          ld a,e
2201  B7          or a
2202  CC 46 22    call z,2246       ...
2205  DD 73 10    ld (ix+10),e
2208  F3          di
2209  DD 71 06    ld (ix+06),c
220C  DD 36 07 80 ld (ix+07),80     =128.
2210  FB          ei
2211  B7          or a
2212  C9          ret

2213  57          ld d,a
2214  4B          ld c,e
2215  3E 0D       ld a,0D           =13.
2217  CD 26 08    call 0826         MC SOUND REGISTER, send <a>=reg#, <c>=data
221A  4A          ld c,d
221B  3E 0B       ld a,0B           =11.
221D  CD 26 08    call 0826         MC SOUND REGISTER, send <a>=reg#, <c>=data
2220  4E          ld c,(hl)
2221  3E 0C       ld a,0C           =12.
2223  CD 26 08    call 0826         MC SOUND REGISTER, send <a>=reg#, <c>=data
2226  3E 10       ld a,10           =16.
2228  CD 73 22    call 2273         ...
222B  CD 46 22    call 2246         ...
222E  7B          ld a,e
222F  3C          inc a
2230  20 8D       jr nz,21BF        ...
2232  21 B2 21    ld hl,21B2        ...
2235  CD 65 22    call 2265         ...
2238  18 85       jr 21BF           ...

223A  AF          xor a
223B  DD 77 03    ld (ix+03),a
223E  DD 77 07    ld (ix+07),a
2241  DD 77 04    ld (ix+04),a
2244  37          scf
2245  C9          ret

      @ 2202! 222B!
2246  DD 35 0C    dec (ix+0C)
2249  20 1E       jr nz,2269        ...
224B  DD 7E 09    ld a,(ix+09)
224E  87          add a,a
224F  21 B2 21    ld hl,21B2        ...
2252  30 11       jr nc,2265        ...
2254  DD 34 08    inc (ix+08)
2257  20 06       jr nz,225F        ...
2259  DD 34 09    inc (ix+09)
225C  1E FF       ld e,FF           =255.
225E  C8          ret z
225F  DD 6E 0A    ld l,(ix+0A)
2262  DD 66 0B    ld h,(ix+0B)
```

```
      @ 21AD! 2235! 2252'
2265  7E          ld a,(hl)
2266  DD 77 0C    ld (ix+0C),a
2269  23          inc hl
226A  5E          ld e,(hl)
226B  23          inc hl
226C  DD 75 0D    ld (ix+0D),l
226F  DD 74 0E    ld (ix+0E),h
2272  C9          ret

      @ 21DA! 2228!
2273  DD 77 0F    ld (ix+0F),a

      @ 1EFA! 22A1!
2276  4F          ld c,a
2277  DD 7E 00    ld a,(ix+00)
227A  C6 08       add a,08          =8.
227C  C3 26 08    jp 0826           MC SOUND REGISTER, send <a>=reg#, <c>=data

      @ 1EAE! 1F54! 20AC
227F  DD 7E 01    ld a,(ix+01)
2282  2F          cpl
2283  21 52 B5    ld hl,B552        SOUND channel bits of active sounds
2286  F3          di
2287  A6          and (hl)
2288  77          ld (hl),a
2289  FB          ei
228A  AF          xor a

      @ 2193!
228B  47          ld b,a
228C  DD 7E 01    ld a,(ix+01)
228F  DD B6 02    or (ix+02)
2292  21 19 B6    ld hl,B619        ...
2295  F3          di
2296  B6          or (hl)
2297  A8          xor b
2298  BE          cp (hl)
2299  77          ld (hl),a
229A  FB          ei
229B  20 03       jr nz,22A0        ...
229D  78          ld a,b
229E  B7          or a
229F  C0          ret nz
22A0  AF          xor a
22A1  CD 76 22    call 2276         ...
22A4  F3          di
22A5  4E          ld c,(hl)
22A6  3E 07       ld a,07           =7.
22A8  C3 26 08    jp 0826           MC SOUND REGISTER, send <a>=reg#, <c>=data

      @ 20E2!
22AB  CD 24 23    call 2324         ...
22AE  7B          ld a,e
22AF  CD 4E 23    call 234E         SOUND get TONE ENV ADDR, <a>=env#, (hl)=addr
22B2  D0          ret nc
22B3  7E          ld a,(hl)
22B4  E6 7F       and 7F            =127.
22B6  C8          ret z
22B7  DD 75 11    ld (ix+11),l
22BA  DD 74 12    ld (ix+12),h
22BD  CD 13 23    call 2313         ...
22C0  18 09       jr 22CB           ...
```

```
      @ 1F24!
22C2  DD 6E 14    ld l,(ix+14)
22C5  DD 66 15    ld h,(ix+15)
22C8  DD 5E 18    ld e,(ix+18)
22CB  4E          ld c,(hl)
22CC  23          inc hl
22CD  7B          ld a,e
22CE  D6 F0       sub F0          =240.
22D0  38 04       jr c,22D6       ...
22D2  1E 00       ld e,00         =0.
22D4  18 0E       jr 22E4         ...

22D6  1D          dec e
22D7  79          ld a,c
22D8  87          add a,a
22D9  9F          sbc a,a
22DA  57          ld d,a
22DB  DD 7E 16    ld a,(ix+16)
22DE  81          add a,c
22DF  4F          ld c,a
22E0  DD 7E 17    ld a,(ix+17)
22E3  8A          adc a,d
22E4  57          ld d,a
22E5  CD 24 23    call 2324       ...
22E8  4E          ld c,(hl)
22E9  7B          ld a,e
22EA  B7          or a
22EB  20 19       jr nz,2306      ...
22ED  DD 7E 13    ld a,(ix+13)
22F0  3D          dec a
22F1  20 10       jr nz,2303      ...
22F3  DD 6E 11    ld l,(ix+11)
22F6  DD 66 12    ld h,(ix+12)
22F9  7E          ld a,(hl)
22FA  C6 80       add a,80        =128.
22FC  38 05       jr c,2303       ...
22FE  DD 36 04 00 ld (ix+04),00   =0.
2302  C9          ret

2303  CD 13 23    call 2313       ...
2306  DD 73 18    ld (ix+18),e
2309  F3          di
230A  DD 71 05    ld (ix+05),c
230D  DD 36 04 80 ld (ix+04),80   =128.
2311  FB          ei
2312  C9          ret

      @ 22BD! 2303!
2313  DD 77 13    ld (ix+13),a
2316  23          inc hl
2317  5E          ld e,(hl)
2318  23          inc hl
2319  DD 75 14    ld (ix+14),l
231C  DD 74 15    ld (ix+15),h
231F  7B          ld a,e
2320  B7          or a
2321  C0          ret nz
2322  1C          inc e
2323  C9          ret

      @ 22AB! 22E5!
2324  DD 7E 00    ld a,(ix+00)
2327  87          add a,a
2328  F5          push af
2329  DD 71 16     ld (ix+16),c
```

```
232C  CD 26 08    call 0826       MC SOUND REGISTER, send <a>=reg#, <c>=data
232F  F1          pop af
2330  3C          inc a
2331  4A          ld c,d
2332  DD 71 17    ld (ix+17),c
2335  C3 26 08    jp 0826         MC SOUND REGISTER, send <a>=reg#, <c>=data

----- SOUND set AMPL ENVELOPE, <a>=env#, (hl)=data
      @ BCBC!
2338  11 0A B6    ld de,B60A      SOUND amplitude envelope
233B  18 03       jr 2340

----- SOUND set TONE ENVELOPE, <a>=env#, (hl)=data
      @ BCBF!
233D  11 FA B6    ld de,B6FA      SOUND tone envelope
2340  EB          ex de,hl
2341  CD 51 23    call 2351       get addr (hl) of envelope <a>, in: (hl)=bloc
2344  EB          ex de,hl
2345  D0          ret nc
2346  ED B0       ldir
2348  C9          ret

----- SOUND get AMPL ENV ADDR, <a>=env#, (hl)=addr
      @ BCC2!
2349  21 0A B6    ld hl,B60A      SOUND amplitude envelope
234C  18 03       jr 2351         get addr (hl) of envelope <a>, in: (hl)=bloc

----- SOUND get TONE ENV ADDR, <a>=env#, (hl)=addr
      @ 22AF! BCC5!
234E  21 FA B6    ld hl,B6FA      SOUND tone envelope

----- get addr (hl) of envelope <a>, in: (hl)=block start
      @ 2341! 234C'
2351  B7          or a
2352  C8          ret z           envelope #0 illegal
2353  FE 10       cp 10           > 15.?
2355  D0          ret nc          it's too big
2356  01 10 00    ld bc,0010      len of one entry
2359  87          add a,a         *2
235A  87          add a,a         *4
235B  87          add a,a         *8
235C  87          add a,a         *16
235D  85          add a,l
235E  6F          ld l,a          + start address
235F  8C          adc a,h
2360  95          sub l
2361  67          ld h,a
2362  37          scf
2363  C9          ret

2364  C7 C7 C7 C7 C7 C7 C7 C7  C7 C7 C7 C7
```

```
----- CAS INITIALISE cassette manager
      @ 064C! BC65!
2370  CD 01 24     call 2401          CAS IN ABANDON
2373  CD 2E 24     call 242E          CAS OUT ABANDON
2376  AF           xor a              enable
2377  CD 8E 23     call 238E          CAS NOISY, enable or disable prompt messages
237A  21 4D 01     ld hl,014D         len of half a zero bit
237D  3E 19        ld a,19            precompensation

----- CAS SET write SPEED, <hl>=len of half a zero bit, <a>=precompensation
      @ BC68!
237F  29           add hl,hl
2380  29           add hl,hl
2381  29           add hl,hl
2382  29           add hl,hl
2383  29           add hl,hl
2384  29           add hl,hl
2385  0F           rrca
2386  0F           rrca
2387  E6 3F        and 3F             mask out bit 6+7
2389  6F           ld l,a
238A  22 D1 B8     ld (B8D1),hl       CAS write speed
238D  C9           ret

----- CAS NOISY, enable or disable prompt messages <a>
      @ 2377! 2534! BC6B!
238E  32 00 B8     ld (B800),a        CAS IN flag; enable prompt message
2391  C9           ret

----- CAS IN OPEN, (hl)=filename, <b>=len, (de)=2kbuff
      @ BC77!
2392  DD 21 02 B8 ld ix,B802          CAS IN file type on read
2396  CD AF 23     call 23AF          ...
2399  D0           ret nc
239A  E5           push hl
239B  CD 3F 25      call 253F         ...
239E  ED 5B 1C B8   ld de,(B81C)      CAS IN, data location
23A2  ED 4B 1F B8   ld bc,(B81F)      CAS IN, user fields
23A6  3A 19 B8      ld a,(B819)       CAS IN file type
23A9  E1           pop hl
23AA  C9           ret

----- CAS OUT OPEN, (hl)=filename, <b>=len, (de)=2kbuff
      @ BC8C!
23AB  DD 21 47 B8 ld ix,B847          CAS OUT DIRECT file type on write

      @ 2396!
23AF  DD 7E 00     ld a,(ix+00)       file type on write
23B2  B7           or a
23B3  C0           ret nz
23B4  DD E5        push ix
23B6  E3            ex (sp),hl
23B7  36 01         ld (hl),01        =1.
23B9  23            inc hl
23BA  73            ld (hl),e
23BB  23            inc hl
23BC  72            ld (hl),d
23BD  23            inc hl
23BE  73            ld (hl),e
23BF  23            inc hl
23C0  72            ld (hl),d
23C1  23            inc hl
23C2  EB            ex de,hl
23C3  E1           pop hl
23C4  D5           push de
```

```
23C5  0E 40        ld c,40          =64.
23C7  12           ld (de),a
23C8  13           inc de
23C9  0D           dec c
23CA  20 FB        jr nz,23C7       ...
23CC  D1          pop de
23CD  D5          push de
23CE  78           ld a,b
23CF  FE 10        cp 10            =16.
23D1  38 02        jr c,23D5        ...
23D3  06 10        ld b,10          =16.
23D5  04           inc b
23D6  48           ld c,b
23D7  18 07        jr 23E0          ...

23D9  E7          rst 4             = ld(hl) RAM
23DA  23          inc hl
23DB  CD B6 27    call 27B6         CAS change <a> to upper case letter
23DE  12          ld (de),a
23DF  13          inc de
23E0  10 F7       djnz 23D9         ...
23E2  0D          dec c
23E3  28 09       jr z,23EE         ...
23E5  1B          dec de
23E6  1A          ld a,(de)
23E7  EE 20       xor 20            =32.
23E9  20 03       jr nz,23EE        ...
23EB  12          ld (de),a
23EC  18 F4       jr 23E2           ...

23EE  E1          pop hl
23EF  DD 36 15 01 ld (ix+15),01     =1.
23F3  DD 36 17 16 ld (ix+17),16     =22.
23F7  DD 35 1C    dec (ix+1C)
23FA  37          scf
23FB  C9          ret

----- CAS IN CLOSE
      @ BC7A!
23FC  3A 02 B8    ld a,(B802)       CAS IN file type on read
23FF  B7          or a
2400  C8          ret z

----- CAS IN ABANDON
      @ 2370! 253C  BC7D!
2401  21 02 B8    ld hl,B802        CAS IN file type on read
2404  3E 01       ld a,01           =1.
2406  36 00       ld (hl),00        =0.
2408  23          inc hl
2409  5E          ld e,(hl)
240A  23          inc hl
240B  56          ld d,(hl)
240C  21 CC B8    ld hl,B8CC        ...
240F  AE          xor (hl)
2410  37          scf
2411  C0          ret nz
2412  77          ld (hl),a
2413  9F          sbc a,a
2414  C9          ret

----- CAS OUT CLOSE
      @ BC8F!
2415  3A 47 B8    ld a,(B847)       CAS OUT DIRECT file type on write
2418  FE 04       cp 04             =4.
241A  28 12       jr z,242E         CAS OUT ABANDON
```

```
241C  C6 FF       add a,FF          =255.
241E  D0          ret nc
241F  21 5D B8    ld hl,B85D        CAS OUT DIRECT, last block flag
2422  36 FF       ld (hl),FF        =255.
2424  23          inc hl
2425  23          inc hl
2426  7E          ld a,(hl)
2427  23          inc hl
2428  B6          or (hl)
2429  37          scf
242A  C4 14 26    call nz,2614      ...
242D  D0          ret nc

----- CAS OUT ABANDON
      @ 2373! 241A' BC92!
242E  21 47 B8    ld hl,B847        CAS OUT DIRECT file type on write
2431  3E 02       ld a,02           =2.
2433  18 D1       jr 2406           ...

----- CAS IN CHAR from input file
      @ 2496! BC80!
2435  E5          push hl
2436  D5           push de
2437  C5            push bc
2438  06 02           ld b,02       file type
243A  CD 8B 24        call 248B     ...
243D  20 1A           jr nz,2459    illegal type access
243F  2A 1A B8        ld hl,(B81A)  CAS IN, data length
2442  7C              ld a,h
2443  B5              or l
2444  37              scf
2445  CC 3F 25        call z,253F   ...
2448  30 0F           jr nc,2459    EOF found, return
244A  2A 1A B8        ld hl,(B81A)  CAS IN, data length
244D  2B              dec hl
244E  22 1A B8        ld (B81A),hl  CAS IN, data length
2451  2A 05 B8        ld hl,(B805)  CAS IN buffer pointer (hi)
2454  E7              rst 4         get the char from buffer
2455  23              inc hl
2456  22 05 B8        ld (B805),hl  CAS IN buffer pointer (hi)
2459  18 2C           jr 2487

----- CAS OUT CHAR <a> to output file
      @ BC95!
245B  E5          push hl
245C  D5           push de
245D  C5            push bc
245E  4F              ld c,a
245F  21 47 B8        ld hl,B847    CAS OUT DIRECT file type on write
2462  06 02           ld b,02       =2.
2464  CD 8E 24        call 248E     ...
2467  20 1E           jr nz,2487    ...
2469  2A 5F B8        ld hl,(B85F)  CAS OUT, len of data, updated while writing
246C  11 00 08        ld de,0800    ...
246F  ED 52           sbc hl,de
2471  C5              push bc
2472  D4 14 26         call nc,2614 ...
2475  C1              pop bc
2476  30 0F           jr nc,2487    ...
2478  2A 5F B8        ld hl,(B85F)  CAS OUT, len of data, updated while writing
247B  23              inc hl
247C  22 5F B8        ld (B85F),hl  CAS OUT, len of data, updated while writing
247F  2A 4A B8        ld hl,(B84A)  CAS OUT DIRECT, pointer to data (hi)
2482  71              ld (hl),c
2483  23              inc hl
```

```
2484  22 4A B8       ld (B84A),hl  CAS OUT DIRECT, pointer to data (hi)
2487  C1             pop bc
2488  D1             pop de
2489  E1             pop hl
248A  C9             ret

      @ 243A! 24AE!
248B  21 02 B8       ld hl,B802        CAS IN file type on read

      @ 2464! 24F2!
248E  7E             ld a,(hl)
248F  B8             cp b
2490  C8             ret z
2491  EE 01          xor 01            =1.
2493  C0             ret nz
2494  70             ld (hl),b
2495  C9             ret

----- CAS TEST EOF
      @ BC89!
2496  CD 35 24       call 2435         CAS IN CHAR from input file
2499  D0             ret nc

----- CAS RETURN, put last char read back
      @ BC86!
249A  E5             push hl
249B  2A 1A B8       ld hl,(B81A)      CAS IN, data length
249E  23             inc hl
249F  22 1A B8       ld (B81A),hl      CAS IN, data length
24A2  2A 05 B8       ld hl,(B805)      CAS IN buffer pointer (hi)
24A5  2B             dec hl
24A6  22 05 B8       ld (B805),hl      CAS IN buffer pointer (hi)
24A9  E1             pop hl
24AA  C9             ret

----- CAS IN DIRECT, read input file into store (hl)
      @ BC83!
24AB  EB             ex de,hl
24AC  06 03          ld b,03           =3.
24AE  CD 8B 24       call 248B         ...
24B1  C0             ret nz
24B2  ED 53 1C B8    ld (B81C),de      CAS IN, data location
24B6  CD CF 24       call 24CF         ...
24B9  2A 1C B8       ld hl,(B81C)      CAS IN, data location
24BC  ED 5B 1A B8    ld de,(B81A)      CAS IN, data length
24C0  19             add hl,de
24C1  22 1C B8       ld (B81C),hl      CAS IN, data location
24C4  CD 3F 25       call 253F         ...
24C7  38 F0          jr c,24B9         ...
24C9  C8             ret z
24CA  2A A6 B8       ld hl,(B8A6)      CAS OUT, user fields (entry addr for machine
24CD  37             scf
24CE  C9             ret

      @ 24B6!
24CF  2A 03 B8       ld hl,(B803)      CAS IN buffer pointer (lo)
24D2  ED 5B 1C B8    ld de,(B81C)      CAS IN, data location
24D6  ED 4B 1A B8    ld bc,(B81A)      CAS IN, data length
24DA  7B             ld a,e
24DB  95             sub l
24DC  7A             ld a,d
24DD  9C             sbc a,h
24DE  DA A6 BA       jp c,BAA6         KL ldir, ROMs disabled
24E1  09             add hl,bc
24E2  2B             dec hl
```

```
24E3  EB          ex de,hl
24E4  09          add hl,bc
24E5  2B          dec hl
24E6  EB          ex de,hl
24E7  C3 AC BA    jp BAAC            KL lddr, ROMs disabled

----- CAS OUT DIRECT, (hl)=data, <de>=len, <a>=type, (bc)=entry addr header
      @ BC98!
24EA  E5          push hl
24EB  C5           push bc
24EC  4F            ld c,a
24ED  21 47 B8      ld hl,B847       CAS OUT DIRECT file type on write
24F0  06 03         ld b,03          =3.
24F2  CD 8E 24      call 248E        ...
24F5  79            ld a,c
24F6  C1           pop bc
24F7  E1          pop hl
24F8  C0          ret nz
24F9  32 5E B8    ld (B85E),a        CAS OUT DIRECT, file type
24FC  ED 53 64 B8 ld (B864),de       CAS OUT DIRECT, total len of data
2500  ED 43 66 B8 ld (B866),bc       CAS OUT DIRECT; entry for HEADER
2504  22 48 B8    ld (B848),hl       CAS OUT DIRECT, pointer to data (lo)
2507  ED 53 5F B8 ld (B85F),de       CAS OUT, len of data, updated while writing
250B  21 FF F7    ld hl,F7FF         = -2K
250E  19          add hl,de
250F  3F          ccf
2510  D8          ret c
2511  21 00 08    ld hl,0800         = 2K
2514  22 5F B8    ld (B85F),hl       CAS OUT, len of data, updated while writing
2517  EB          ex de,hl
2518  ED 52       sbc hl,de
251A  E5          push hl
251B  2A 48 B8     ld hl,(B848)      CAS OUT DIRECT, pointer to data (lo)
251E  19           add hl,de
251F  E5           push hl
2520  CD 14 26      call 2614        ...
2523  E1           pop hl
2524  D1          pop de
2525  D0          ret nc
2526  18 DC       jr 2504            ...

----- CAS CATALOG, (de)= 2k buffer to use
      @ BC9B!
2528  21 02 B8    ld hl,B802         CAS IN file type on read
252B  7E          ld a,(hl)
252C  B7          or a
252D  C0          ret nz
252E  36 05       ld (hl),05         mark for CAT access
2530  ED 53 03 B8 ld (B803),de       CAS IN buffer pointer (lo)
2534  CD 8E 23    call 238E          CAS NOISY, enable or disable prompt messages
2537  CD 44 25    call 2544          ...
253A  38 FB       jr c,2537          ...
253C  C3 01 24    jp 2401            CAS IN ABANDON

      @ 239B! 2445! 24C4!
253F  3A 18 B8    ld a,(B818)        CAS IN last block flag
2542  B7          or a
2543  C0          ret nz

      @ 2537!
2544  01 01 83    ld bc,8301         ...
2547  CD 73 26    call 2673          ...
254A  30 5C       jr nc,25A8         ...
```

```
        @ 25B3' 25BC' 25C3'
254C  21 8C B8     ld hl,B88C        CAS OUT filename   HEADER RECORD up to B8CB
254F  11 40 00     ld de,0040        ...
2552  3E 2C        ld a,2C           =44.
2554  CD 36 28     call 2836         CAS READ a record, (hl)=data, <de>=len, <a>=
2557  30 4F        jr nc,25A8        ...
2559  CD C5 25     call 25C5         ...
255C  20 57        jr nz,25B5        ...
255E  06 8B        ld b,8B           =139.
2560  38 02        jr c,2564         ...
2562  06 89        ld b,89           =137.
2564  CD 92 26     call 2692         ...
2567  ED 5B 9F B8  ld de,(B89F)      CAS OUT, data length
256B  2A 1C B8     ld hl,(B81C)      CAS IN, data location
256E  3A 02 B8     ld a,(B802)       CAS IN file type on read
2571  FE 03        cp 03             =3.
2573  28 0E        jr z,2583         ...
2575  21 FF F7     ld hl,F7FF        ...
2578  19           add hl,de
2579  3E 04        ld a,04           =4.
257B  38 2B        jr c,25A8         ...
257D  2A 03 B8     ld hl,(B803)      CAS IN buffer pointer (lo)
2580  22 05 B8     ld (B805),hl      CAS IN buffer pointer (hi)
2583  3E 16        ld a,16           =22.
2585  CD 36 28     call 2836         CAS READ a record, (hl)=data, <de>=len, <a>=
2588  30 1E        jr nc,25A8        ...
258A  21 17 B8     ld hl,B817        CAS IN block number
258D  34           inc (hl)
258E  3A 9D B8     ld a,(B89D)       CAS OUT, file type
2591  23           inc hl
2592  77           ld (hl),a
2593  AF           xor a
2594  32 1E B8     ld (B81E),a       CAS IN, first block flag
2597  2A 9F B8     ld hl,(B89F)      CAS OUT, data length
259A  22 1A B8     ld (B81A),hl      CAS IN, data length
259D  CD BF 27     call 27BF         CAS get file type on read; cp 05
25A0  3E 8C        ld a,8C           =140.
25A2  CC 0C 27     call z,270C       ...
25A5  37           scf
25A6  18 65        jr 260D           ...

        @ 254A' 2557' 257B' 2588'
25A8  B7           or a
25A9  21 02 B8     ld hl,B802        CAS IN file type on read
25AC  28 5D        jr z,260B         ...
25AE  06 85        ld b,85           =133.
25B0  CD 13 27     call 2713         ...
25B3  18 97        jr 254C           ...

25B5  F5           push af
25B6  06 88         ld b,88          =136.
25B8  CD 92 26      call 2692        ...
25BB  F1           pop af
25BC  30 8E        jr nc,254C        ...
25BE  06 87        ld b,87           =135.
25C0  CD 11 27     call 2711         ...
25C3  18 87        jr 254C           ...

        @ 2559!
25C5  CD BF 27     call 27BF         CAS get file type on read; cp 05
25C8  37           scf
25C9  C8           ret z
25CA  3A 1E B8     ld a,(B81E)       CAS IN, first block flag
25CD  B7           or a
25CE  28 1B        jr z,25EB         ...
```

```
25D0  3A A3 B8    ld a,(B8A3)       CAS OUT, first block flag
25D3  2F          cpl
25D4  B7          or a
25D5  C0          ret nz
25D6  3A 07 B8    ld a,(B807)       CAS IN filename    HEADER RECORD up to B846
25D9  B7          or a
25DA  C4 F3 25    call nz,25F3      ...
25DD  C0          ret nz
25DE  21 8C B8    ld hl,B88C        CAS OUT filename   HEADER RECORD up to B8CB
25E1  11 07 B8    ld de,B807        CAS IN filename    HEADER RECORD up to B846
25E4  01 40 00    ld bc,0040        ...
25E7  ED B0       ldir
25E9  AF          xor a
25EA  C9          ret

25EB  CD F3 25    call 25F3         ...
25EE  C0          ret nz
25EF  EB          ex de,hl
25F0  1A          ld a,(de)
25F1  BE          cp (hl)
25F2  C9          ret

      @ 25DA! 25EB!
25F3  21 07 B8    ld hl,B807        CAS IN filename    HEADER RECORD up to B846
25F6  11 8C B8    ld de,B88C        CAS OUT filename   HEADER RECORD up to B8CB
25F9  06 10       ld b,10           =16.
25FB  1A          ld a,(de)
25FC  CD B6 27    call 27B6         CAS change <a> to upper case letter
25FF  4F          ld c,a
2600  7E          ld a,(hl)
2601  CD B6 27    call 27B6         CAS change <a> to upper case letter
2604  A9          xor c
2605  C0          ret nz
2606  23          inc hl
2607  13          inc de
2608  10 F1       djnz 25FB         ...
260A  C9          ret

      @ 25AC' 266A'
260B  36 04       ld (hl),04        =4.

      @ 25A6' 2664'
260D  9F          sbc a,a
260E  F5          push af
260F  CD 4F 2A     call 2A4F        CAS STOP MOTOR
2612  F1          pop af
2613  C9          ret

      @ 242A! 2472! 2520!
2614  01 02 84    ld bc,8402        ...
2617  CD 73 26    call 2673         ...
261A  30 4A       jr nc,2666        ...
261C  06 8A       ld b,8A           =138.
261E  11 4C B8    ld de,B84C        CAS OUT DIRECT filename   HEADER RECORD up t
2621  CD 95 26    call 2695         ...
2624  21 63 B8    ld hl,B863        CAS OUT DIRECT, first block flag
2627  CD 88 26    call 2688         ...
262A  30 3A       jr nc,2666        ...
262C  2A 48 B8    ld hl,(B848)      CAS OUT DIRECT, pointer to data (lo)
262F  22 4A B8    ld (B84A),hl      CAS OUT DIRECT, pointer to data (hi)
2632  22 61 B8    ld (B861),hl      CAS OUT DIRECT, data location
2635  E5          push hl
2636  21 4C B8     ld hl,B84C       CAS OUT DIRECT filename   HEADER RECORD up t
2639  11 40 00     ld de,0040       ...
263C  3E 2C        ld a,2C          =44.
```

```
263E  CD 3F 28     call 283F        CAS WRITE a record, (hl)=data, <de>=len, <a>
2641  E1           pop hl
2642  30 22        jr nc,2666       ...
2644  ED 5B 5F B8  ld de,(B85F)     CAS OUT, len of data, updated while writing
2648  3E 16        ld a,16          =22.
264A  CD 3F 28     call 283F        CAS WRITE a record, (hl)=data, <de>=len, <a>
264D  21 5D B8     ld hl,B85D       CAS OUT DIRECT, last block flag
2650  DC 88 26     call c,2688      ...
2653  30 11        jr nc,2666       ...
2655  21 00 00     ld hl,0000
2658  22 5F B8     ld (B85F),hl     CAS OUT, len of data, updated while writing
265B  21 5C B8     ld hl,B85C       CAS OUT DIRECT block number
265E  34           inc (hl)
265F  AF           xor a
2660  32 63 B8     ld (B863),a      CAS OUT DIRECT, first block flag
2663  37           scf
2664  18 A7        jr 260D          ...

      @ 261A' 262A' 2642' 2653'
2666  B7           or a
2667  21 47 B8     ld hl,B847       CAS OUT DIRECT file type on write
266A  28 9F        jr z,260B        ...
266C  06 86        ld b,86          =134.
266E  CD 13 27     call 2713        ...
2671  18 B9        jr 262C          ...

      @ 2547! 2617!
2673  21 CC B8     ld hl,B8CC       ...
2676  79           ld a,c
2677  BE           cp (hl)
2678  36 00        ld (hl),00       =0.
267A  37           scf
267B  E5           push hl
267C  C5            push bc
267D  C4 60 27       call nz,2760   ...
2680  C1            pop bc
2681  E1           pop hl
2682  9F           sbc a,a
2683  D0           ret nc
2684  71           ld (hl),c
2685  C3 4B 2A     jp 2A4B          CAS START MOTOR

      @ 2627! 2650!
2688  7E           ld a,(hl)
2689  B7           or a
268A  37           scf
268B  C8           ret z
268C  01 2C 01     ld bc,012C       ...
268F  C3 72 2A     jp 2A72          ...

      @ 2564! 25B8!
2692  11 8C B8     ld de,B88C       CAS OUT filename   HEADER RECORD up to B8CB

      @ 2621!
2695  3A 00 B8     ld a,(B800)      CAS IN flag; enable prompt message
2698  B7           or a
2699  C0           ret nz
269A  32 01 B8     ld (B801),a      CAS IN flag ??
269D  CD 83 27     call 2783        set cursor to column 1
26A0  CD 26 27     call 2726        ...
26A3  1A           ld a,(de)
26A4  B7           or a
26A5  20 0A        jr nz,26B1       ...
26A7  3E 8E        ld a,8E          =142.
26A9  CD 27 27     call 2727        ...
```

```
26AC  01 10 00    ld bc,0010
26AF  18 2E       jr 26DF          ...

26B1  CD BF 27    call 27BF        CAS get file type on read; cp 05
26B4  01 00 10    ld bc,1000       ...
26B7  28 0D       jr z,26C6        ...
26B9  6B          ld l,e
26BA  62          ld h,d
26BB  7E          ld a,(hl)
26BC  B7          or a
26BD  28 04       jr z,26C3        ...
26BF  0C          inc c
26C0  23          inc hl
26C1  10 F8       djnz 26BB        ...
26C3  78          ld a,b
26C4  41          ld b,c
26C5  4F          ld c,a
26C6  CD 8D 27    call 278D        ...
26C9  1A          ld a,(de)
26CA  CD B6 27    call 27B6        CAS change <a> to upper case letter
26CD  B7          or a
26CE  20 02       jr nz,26D2       ...
26D0  3E 20       ld a,20          =32.
26D2  C5          push bc
26D3  D5           push de
26D4  CD 34 13      call 1334      TXT WRITE char <a> to screen
26D7  D1           pop de
26D8  C1          pop bc
26D9  13          inc de
26DA  10 ED       djnz 26C9        ...
26DC  CD 5C 27    call 275C        ...
26DF  EB          ex de,hl
26E0  09          add hl,bc
26E1  EB          ex de,hl
26E2  3E 8D       ld a,8D          =141.
26E4  CD 27 27    call 2727        ...
26E7  06 02       ld b,02          =2.
26E9  CD 8D 27    call 278D        ...
26EC  1A          ld a,(de)
26ED  CD A4 27    call 27A4        ...
26F0  CD 5C 27    call 275C        ...
26F3  13          inc de
26F4  CD BF 27    call 27BF        CAS get file type on read; cp 05
26F7  20 0B       jr nz,2704       ...
26F9  13          inc de
26FA  1A          ld a,(de)
26FB  E6 0F       and 0F           =15.
26FD  C6 24       add a,24         =36.
26FF  CD 80 27    call 2780        TXT OUTPUT char or ctl code <a> to VDU
2702  18 58       jr 275C          ...

2704  1A          ld a,(de)
2705  21 01 B8    ld hl,B801       CAS IN flag ??
2708  B6          or (hl)
2709  C8          ret z
270A  18 6F       jr 277B          set cursor to col 1, print linefeed

      @ 25A2!
270C  CD 27 27    call 2727        ...
270F  18 6A       jr 277B          set cursor to col 1, print linefeed
```

```
      @ 25C0!
2711  3E FF       ld a,FF          =255.

      @ 25B0! 266E!
2713  F5          push af
2714  CD 1F 27     call 271F       ...
2717  F1          pop af
2718  C6 60       add a,60         =96.
271A  D4 80 27    call nc,2780     TXT OUTPUT char or ctl code <a> to VDU
271D  18 5C       jr 277B          set cursor to col 1, print linefeed

      @ 2714! 2766!
271F  CD 80 11    call 1180        TXT GET CURSOR position (hl), roll count <a>
2722  25          dec h
2723  C4 7B 27    call nz,277B     set cursor to col 1, print linefeed

      @ 26A0!
2726  78          ld a,b

      @ 26A9! 26E4! 270C! 2743
2727  E5          push hl
2728  E6 7F        and 7F          =127.
272A  47           ld b,a
272B  21 C5 27     ld hl,27C5      'press play then any key
272E  28 07        jr z,2737       ...
2730  7E           ld a,(hl)
2731  23           inc hl
2732  B7           or a
2733  20 FB        jr nz,2730      ...
2735  10 F9        djnz 2730       ...
2737  7E           ld a,(hl)
2738  B7           or a
2739  28 05        jr z,2740       ...
273B  CD 43 27     call 2743       ...
273E  18 F7        jr 2737         ...

2740  E1          pop hl
2741  23          inc hl
2742  C9          ret

      @ 273B!
2743  FA 27 27    jp m,2727        ...
2746  E5          push hl
2747  06 00        ld b,00         =0.
2749  04           inc b
274A  7E           ld a,(hl)
274B  23           inc hl
274C  07           rlca
274D  30 FA        jr nc,2749      ...
274F  CD 8D 27     call 278D       ...
2752  E1          pop hl
2753  7E          ld a,(hl)
2754  23          inc hl
2755  E6 7F       and 7F           =127.
2757  CD 80 27    call 2780        TXT OUTPUT char or ctl code <a> to VDU
275A  10 F7       djnz 2753        ...

      @ 26DC! 26F0! 2702'
275C  3E 20       ld a,20          'SPACE
275E  18 20       jr 2780          TXT OUTPUT char or ctl code <a> to VDU
```

```
     @ 267D!
2760  3A 00 B8    ld a,(B800)       CAS IN flag; enable prompt message
2763  B7          or a
2764  37          scf
2765  C0          ret nz
2766  CD 1F 27    call 271F         ...
2769  CD 42 1A    call 1A42         KM READ CHAR from keyboard =<a>
276C  38 FB       jr c,2769         ...
276E  CD 79 12    call 1279         TXT CURSOR ON
2771  CD 56 1B    call 1B56         KM WAIT for KEY
2774  CD 81 12    call 1281         TXT CURSOR OFF
2777  FE 1B       cp 1B             'ESC
2779  C8          ret z
277A  37          scf

----- set cursor to col 1, print linefeed
     @ 270A' 270F' 271D' 2723! 27A2'
277B  CD 83 27    call 2783         set cursor to column 1
277E  3E 0A       ld a,0A           'LF  (^J)

----- TXT OUTPUT char or ctl code <a> to VDU
     @ 26FF! 271A! 2757! 275E' 27B4'
2780  C3 00 14    jp 1400           TXT OUTPUT char or ctl code <a> to VDU

----- set cursor to column 1
     @ 269D! 277B!
2783  F5          push af
2784  E5           push hl
2785  3E 01         ld a,01         column 1
2787  CD 5E 11      call 115E       TXT SET cursor to COLUMN <a>
278A  E1           pop hl
278B  F1          pop af
278C  C9          ret

     @ 26C6! 26E9! 274F!
278D  D5          push de
278E  CD 56 12     call 1256        TXT GET WINDOW size, <hl>=left top, <de>=rig
2791  5C           ld e,h
2792  CD 80 11     call 1180        TXT GET CURSOR position (hl), roll count <a>
2795  7C           ld a,h
2796  3D           dec a
2797  83           add a,e
2798  80           add a,b
2799  3D           dec a
279A  BA           cp d
279B  D1          pop de
279C  D8          ret c
279D  3E FF       ld a,FF           set flag
279F  32 01 B8    ld (B801),a       CAS IN flag ??
27A2  18 D7       jr 277B           set cursor to col 1, print linefeed

     @ 26ED! 27B0!
27A4  06 FF       ld b,FF           'IGNORE
27A6  04          inc b
27A7  D6 0A       sub 0A            'LF  (^J)
27A9  30 FB       jr nc,27A6        ...
27AB  C6 3A       add a,3A          ':
27AD  F5          push af
27AE  78           ld a,b
27AF  B7           or a
27B0  C4 A4 27     call nz,27A4     ...
27B3  F1          pop af
27B4  18 CA       jr 2780           TXT OUTPUT char or ctl code <a> to VDU
```

```
----- CAS change <a> to upper case letter
      @ 23DB! 25FC! 2601! 26CA!
27B6  FE 61        cp 61           'a
27B8  D8           ret c
27B9  FE 7B        cp 7B           '{
27BB  D0           ret nc
27BC  C6 E0        add a,E0        -20
27BE  C9           ret

----- CAS get file type on read; cp 05
      @ 259D! 25C5! 26B1! 26F4!
27BF  3A 02 B8     ld a,(B802)     CAS IN file type on read
27C2  FE 05        cp 05           =5.
27C4  C9           ret

----- 'press play then any key
      @ 272B:
27C5  50 72 65 73 F3 00                                          'Press.
27CB  50 4C 41 D9 74 68 65 EE  61 6E F9 6B 65 79 BA 00           'PLAYthenanykey:.
27DB  65 72 72 6F F2 00                                          'error.
27E1  80 81 00                                                   '...
27E4  80 52 45 C3 61 6E E4 81  00                                '.RECand..
27ED  52 65 61 E4 82 00                                          'Read..
27F3  57 72 69 74 E5 82 00                                       'Write..
27FA  52 65 77 69 6E E4 74 61  70 E5 00                          'Rewindtape.
2805  46 6F 75 6E 64 20 A0 00                                    'Found  .
280D  4C 6F 61 64 69 6E E7 00                                    'Loading.
2815  53 61 76 69 6E E7 00                                       'Saving.
281C  00                                                         '.
281D  4F EB 00                                                   'Ok.
2820  62 6C 6F 63 EB 00                                          'block.
2826  55 6E 6E 61 6D 65 E4 66  69 6C 65 20 20 20 A0 00           'Unnamedfile

----- CAS READ a record, (hl)=data, <de>=len, <a>=expected sync
      @ 2554! 2585! BCA1!
2836  CD 73 28     call 2873       ...
2839  F5           push af
283A  21 B8 28      ld hl,28B8     ...
283D  18 19         jr 2858        ...

----- CAS WRITE a record, (hl)=data, <de>=len, <a>=sync char
      @ 263E! 264A! BC9E!
283F  CD 73 28     call 2873       ...
2842  F5           push af
2843  CD 64 29      call 2964      ...
2846  21 F7 28      ld hl,28F7     ...
2849  DC 9D 28      call c,289D    ...
284C  DC 79 29      call c,2979    ...
284F  18 0F         jr 2860        ...

----- CAS CHECK tape with store, (hl)=data, <de>=len, <a>=sync char
      @ BCA4!
2851  CD 73 28     call 2873       ...
2854  F5           push af
2855  21 C7 28      ld hl,28C7     ...

      @ 283D'
2858  E5            push hl
2859  CD 19 29       call 2919     ...
285C  E1            pop hl
285D  DC 9D 28      call c,289D    ...
```

```
      @ 284F'
2860  D1          pop de
2861  F5          push af
2862  01 82 F7     ld bc,F782       ...
2865  ED 49        out (c),c
2867  01 10 F6     ld bc,F610       ...
286A  ED 49        out (c),c
286C  FB           ei
286D  7A           ld a,d
286E  CD 51 2A     call 2A51        CAS RESTORE MOTOR to previous state <a>
2871  F1          pop af
2872  C9          ret

      @ 2836! 283F! 2851!
2873  32 CD B8    ld (B8CD),a       ...
2876  1B          dec de
2877  1C          inc e
2878  E5          push hl
2879  D5           push de
287A  CD 68 1E      call 1E68       SOUND RESET
287D  D1           pop de
287E  DD E1       pop ix
2880  CD 4B 2A    call 2A4B         CAS START MOTOR
2883  F3          di
2884  01 0E F4    ld bc,F40E        ...
2887  ED 49       out (c),c
2889  01 D0 F6    ld bc,F6D0        ...
288C  ED 49       out (c),c
288E  0E 10       ld c,10           =16.
2890  ED 49       out (c),c
2892  01 92 F7    ld bc,F792        ...
2895  ED 49       out (c),c
2897  01 58 F6    ld bc,F658        ...
289A  ED 49       out (c),c
289C  C9          ret

      @ 2849! 285D!
289D  7A          ld a,d
289E  B7          or a
289F  28 0D       jr z,28AE         ...
28A1  E5          push hl
28A2  D5           push de
28A3  1E 00         ld e,00         =0.
28A5  CD AE 28      call 28AE       ...
28A8  D1           pop de
28A9  E1          pop hl
28AA  D0          ret nc
28AB  15          dec d
28AC  20 F3       jr nz,28A1        ...

      @ 289F' 28A5!
28AE  01 FF FF    ld bc,FFFF        ...
28B1  ED 43 D3 B8 ld (B8D3),bc      ...
28B5  16 01       ld d,01           =1.
28B7  E9          jp (hl)

      @ 283A: 28C3'
28B8  CD B0 29    call 29B0         ...
28BB  D0          ret nc
28BC  DD 77 00    ld (ix+00),a
28BF  DD 23       inc ix
28C1  15          dec d
28C2  1D          dec e
28C3  20 F3       jr nz,28B8        ...
28C5  18 12       jr 28D9           ...
```

```
        @ 2855: 28D7'
28C7    CD B0 29      call 29B0          ...
28CA    D0            ret nc
28CB    47            ld b,a
28CC    CD DC BA      call BADC          ...
28CF    A8            xor b
28D0    3E 03         ld a,03            =3.
28D2    C0            ret nz
28D3    DD 23         inc ix
28D5    15            dec d
28D6    1D            dec e
28D7    20 EE         jr nz,28C7         ...

        @ 28C5' 28E0'
28D9    15            dec d
28DA    28 06         jr z,28E2          ...
28DC    CD B0 29      call 29B0          ...
28DF    D0            ret nc
28E0    18 F7         jr 28D9            ...

        @ 28DA'
28E2    CD A6 29      call 29A6          ...
28E5    CD B0 29      call 29B0          ...
28E8    D0            ret nc
28E9    AA            xor d
28EA    20 07         jr nz,28F3         ...
28EC    CD B0 29      call 29B0          ...
28EF    D0            ret nc
28F0    AB            xor e
28F1    37            scf
28F2    C8            ret z

        @ 28EA'
28F3    3E 02         ld a,02            =2.
28F5    B7            or a
28F6    C9            ret

        @ 2846: 2902'
28F7    CD DC BA      call BADC          ...
28FA    CD F8 29      call 29F8          ...
28FD    D0            ret nc
28FE    DD 23         inc ix
2900    15            dec d
2901    1D            dec e
2902    20 F3         jr nz,28F7         ...

        @ 290C'
2904    15            dec d
2905    28 07         jr z,290E          ...
2907    AF            xor a
2908    CD F8 29      call 29F8          ...
290B    D0            ret nc
290C    18 F6         jr 2904            ...

        @ 2905'
290E    CD A6 29      call 29A6          ...
2911    CD F8 29      call 29F8          ...
2914    D0            ret nc
2915    7B            ld a,e
2916    C3 F8 29      jp 29F8            ...
```

```
      @ 2859! 2921'
2919  D5          push de
291A  CD 23 29     call 2923      ...
291D  D1          pop de
291E  D8          ret c
291F  B7          or a
2920  C8          ret z
2921  18 F6       jr 2919         ...

      @ 291A!
2923  2E 55       ld l,55         =85.
2925  CD CD 29    call 29CD       ...
2928  D0          ret nc
2929  11 00 00    ld de,0000
292C  62          ld h,d

      @ 2937'
292D  CD CD 29    call 29CD       ...
2930  D0          ret nc
2931  EB          ex de,hl
2932  06 00       ld b,00         =0.
2934  09          add hl,bc
2935  EB          ex de,hl
2936  25          dec h
2937  20 F4       jr nz,292D      ...

      @ 294D' 2950'
2939  61          ld h,c
293A  79          ld a,c
293B  92          sub d
293C  4F          ld c,a
293D  9F          sbc a,a
293E  47          ld b,a
293F  EB          ex de,hl
2940  09          add hl,bc
2941  EB          ex de,hl
2942  CD CD 29    call 29CD       ...
2945  D0          ret nc
2946  7A          ld a,d
2947  CB 3F       srl a
2949  CB 3F       srl a
294B  8A          adc a,d
294C  94          sub h
294D  38 EA       jr c,2939       ...
294F  91          sub c
2950  38 E7       jr c,2939       ...
2952  7A          ld a,d
2953  1F          rra
2954  8A          adc a,d
2955  67          ld h,a
2956  22 CE B8    ld (B8CE),hl    ...
2959  CD B0 29    call 29B0       ...
295C  D0          ret nc
295D  21 CD B8    ld hl,B8CD      ...
2960  AE          xor (hl)
2961  C0          ret nz
2962  37          scf
2963  C9          ret

      @ 2843!
2964  CD 89 2A    call 2A89       ...
2967  21 01 08    ld hl,0801      ...
296A  CD 7C 29    call 297C       ...
296D  D0          ret nc
296E  B7          or a
```

```
296F  CD 08 2A    call 2A08         ...
2972  D0          ret nc
2973  3A CD B8    ld a,(B8CD)       ...
2976  C3 F8 29    jp 29F8           ...

      @ 284C!
2979  21 21 00    ld hl,0021        ...

      @ 296A! 298C'
297C  06 F4       ld b,F4           =244.
297E  ED 78       in a,(c)
2980  E6 04       and 04            =4.
2982  C8          ret z
2983  E5          push hl
2984  37           scf
2985  CD 08 2A     call 2A08        ...
2988  E1          pop hl
2989  2B          dec hl
298A  7C          ld a,h
298B  B5          or l
298C  20 EE       jr nz,297C        ...
298E  37          scf
298F  C9          ret

      @ 29C3! 2A2C!
2990  2A D3 B8    ld hl,(B8D3)      ...
2993  AC          xor h
2994  F2 A0 29    jp p,29A0         ...
2997  7C          ld a,h
2998  EE 08       xor 08            =8.
299A  67          ld h,a
299B  7D          ld a,l
299C  EE 10       xor 10            =16.
299E  6F          ld l,a
299F  37          scf

      @ 2994
29A0  ED 6A       adc hl,hl
29A2  22 D3 B8    ld (B8D3),hl      ...
29A5  C9          ret

      @ 28E2! 290E!
29A6  2A D3 B8    ld hl,(B8D3)      ...
29A9  7D          ld a,l
29AA  2F          cpl
29AB  5F          ld e,a
29AC  7C          ld a,h
29AD  2F          cpl
29AE  57          ld d,a
29AF  C9          ret

      @ 28B8! 28C7! 28DC! 28E5! 28EC! 2959!
29B0  D5          push de
29B1  1E 08        ld e,08          =8.
29B3  2A CE B8     ld hl,(B8CE)     ...
29B6  CD D4 29     call 29D4        ...
29B9  DC DD 29     call c,29DD      ...
29BC  30 0D        jr nc,29CB       ...
29BE  7C           ld a,h
29BF  91           sub c
29C0  9F           sbc a,a
29C1  CB 12        rl d
29C3  CD 90 29     call 2990        ...
29C6  1D           dec e
29C7  20 EA        jr nz,29B3       ...
```

```
29C9  7A           ld a,d
29CA  37           scf
29CB  D1           pop de
29CC  C9           ret

      @ 2925! 292D! 2942!
29CD  06 F4        ld b,F4          =244.
29CF  ED 78        in a,(c)
29D1  E6 04        and 04           =4.
29D3  C8           ret z

      @ 29B6!
29D4  ED 5F        ld a,r
29D6  C6 03        add a,03         =3.
29D8  0F           rrca
29D9  0F           rrca
29DA  E6 1F        and 1F           =31.
29DC  4F           ld c,a

      @ 29B9!
29DD  06 F5        ld b,F5          =245.
29DF  79           ld a,c
29E0  C6 02        add a,02         =2.
29E2  4F           ld c,a
29E3  38 0E        jr c,29F3        ...
29E5  ED 78        in a,(c)
29E7  AD           xor l
29E8  E6 80        and 80           =128.
29EA  20 F3        jr nz,29DF       ...
29EC  AF           xor a
29ED  ED 4F        ld r,a
29EF  CB 0D        rrc l
29F1  37           scf
29F2  C9           ret

      @ 29E3'
29F3  AF           xor a
29F4  ED 4F        ld r,a
29F6  3C           inc a
29F7  C9           ret

      @ 28FA! 2908! 2911! 2916  2976
29F8  D5           push de
29F9  1E 08         ld e,08         =8.
29FB  57            ld d,a
29FC  CB 02         rlc d
29FE  CD 08 2A      call 2A08       ...
2A01  30 03         jr nc,2A06      ...
2A03  1D            dec e
2A04  20 F6         jr nz,29FC      ...
2A06  D1           pop de
2A07  C9           ret

      @ 296F! 2985! 29FE!
2A08  ED 4B D0 B8 ld bc,(B8D0)      ...
2A0C  2A D2 B8     ld hl,(B8D2)     ...
2A0F  9F           sbc a,a
2A10  67           ld h,a
2A11  28 07        jr z,2A1A        ...
2A13  7D           ld a,l
2A14  87           add a,a
2A15  80           add a,b
2A16  6F           ld l,a
2A17  79           ld a,c
2A18  90           sub b
```

```
2A19  4F           ld c,a
2A1A  7D           ld a,l
2A1B  32 D0 B8     ld (B8D0),a      ...
2A1E  2E 0A        ld l,0A          =10.
2A20  CD 37 2A     call 2A37        ...
2A23  38 06        jr c,2A2B        ...
2A25  91           sub c
2A26  30 0C        jr nc,2A34       ...
2A28  2F           cpl
2A29  3C           inc a
2A2A  4F           ld c,a
2A2B  7C           ld a,h
2A2C  CD 90 29     call 2990        ...
2A2F  2E 0B        ld l,0B          =11.
2A31  CD 37 2A     call 2A37        ...
2A34  3E 01        ld a,01          =1.
2A36  C9           ret

      @ 2A20! 2A31!
2A37  ED 5F        ld a,r
2A39  CB 3F        srl a
2A3B  91           sub c
2A3C  30 03        jr nc,2A41       ...
2A3E  3C           inc a
2A3F  20 FD        jr nz,2A3E       ...
2A41  06 F7        ld b,F7          =247.
2A43  ED 69        out (c),l
2A45  F5           push af
2A46  AF            xor a
2A47  ED 4F         ld r,a
2A49  F1           pop af
2A4A  C9           ret

----- CAS START MOTOR
      @ 2685  2880! BC6E!
2A4B  3E 10        ld a,10          set bit 4
2A4D  18 02        jr 2A51          CAS RESTORE MOTOR to previous state <a>

----- CAS STOP MOTOR
      @ 260F! BC71!
2A4F  3E EF        ld a,EF          mask out bit 4

----- CAS RESTORE MOTOR to previous state <a>
      @ 286E! 2A4D' BC74!
2A51  C5           push bc
2A52  06 F6         ld b,F6         =246.
2A54  ED 48         in c,(c)
2A56  04            inc b
2A57  E6 10         and 10          =16.
2A59  3E 08         ld a,08         =8.
2A5B  28 01         jr z,2A5E       ...
2A5D  3C            inc a
2A5E  ED 79         out (c),a
2A60  37            scf
2A61  28 0C         jr z,2A6F       ...
2A63  79            ld a,c
2A64  E6 10         and 10          =16.
2A66  C5            push bc
2A67  01 C8 00       ld bc,00C8     ...
2A6A  37             scf
2A6B  CC 72 2A       call z,2A72    ...
2A6E  C1            pop bc
2A6F  79            ld a,c
2A70  C1           pop bc
2A71  C9           ret
```

```
      @ 268F  2A6B! 2A83'
2A72  C5            push bc
2A73  E5             push hl
2A74  CD 89 2A        call 2A89       ...
2A77  3E 42           ld a,42         =66.
2A79  CD BD 1C        call 1CBD       KM TEST if KEY #<a> is pressed
2A7C  E1             pop hl
2A7D  C1            pop bc
2A7E  20 07         jr nz,2A87        ...
2A80  0B            dec bc
2A81  78            ld a,b
2A82  B1            or c
2A83  20 ED         jr nz,2A72        ...
2A85  37            scf
2A86  C9            ret

2A87  AF            xor a
2A88  C9            ret

      @ 2964! 2A74!
2A89  01 82 06      ld bc,0682        ...
2A8C  0B            dec bc
2A8D  78            ld a,b
2A8E  B1            or c
2A8F  20 FB         jr nz,2A8C        ...
2A91  C9            ret

2A92  C7 C7 C7 C7 C7 C7
```

```
----- EDI LINE EDITOR (hl)
      @ BD3A!
2A98  C5           push bc
2A99  D5            push de
2A9A  E5             push hl
2A9B  E5              push hl
2A9C  01 FF 00         ld bc,00FF   initial count
2A9F  0C               inc c        <c> counts len of string parameter
2AA0  7E               ld a,(hl)    get char from textstring
2AA1  23               inc hl
2AA2  B7               or a
2AA3  20 FA            jr nz,2A9F   not ended yet, take next
2AA5  32 DD B8         ld (B8DD),a  EDI INSERT/OVERWRITE flag
2AA8  CD 6F 2C         call 2C6F    EDI reset COPYCURSOR to zero
2AAB  E1              pop hl
2AAC  CD 67 2D        call 2D67     EDI write string at cursor pos; update curso
2AAF  C5              push bc       <bc>=len of string
2AB0  E5               push hl
2AB1  CD D9 2D          call 2DD9   EDI key-input a char at cursor pos
2AB4  E1               pop hl
2AB5  C1              pop bc
2AB6  CD C6 2A        call 2AC6     EDI handle function key <a>
2AB9  30 F4           jr nc,2AAF    not a legal function key, go back!
2ABB  F5              push af
2ABC  CD D2 2C         call 2CD2    remove old cursor, place at new location
2ABF  F1              pop af
2AC0  E1             pop hl
2AC1  D1            pop de
2AC2  C1           pop bc
2AC3  FE FC        cp FC            'BREAK KEY
2AC5  C9           ret

----- EDI handle function key <a>
      @ 2AB6!
2AC6  E5           push hl
2AC7  21 E0 2A      ld hl,2AE0      table of function key routines
2ACA  5F            ld e,a          save char
2ACB  78            ld a,b
2ACC  B1            or c            test len not zero
2ACD  7B            ld a,e          restore char
2ACE  20 0B         jr nz,2ADB      len not zero
2AD0  FE F0         cp F0           'CURSOR up
2AD2  38 07         jr c,2ADB       but stay within your window
2AD4  FE F4         cp F4           'COPYCU up
2AD6  30 03         jr nc,2ADB      same thing
2AD8  21 1C 2B      ld hl,2B1C      table of cursor functions and 'BEL
2ADB  CD F6 2D      call 2DF6       EDI find key token, address = (hl)
2ADE  E3            ex (sp),hl
2ADF  C9            ret             return to routine found
```

```
----- table of function key routines
2AE0  13 01 2C    2C01 'DC3 (^S)   EDI function key 'DC3 (^S)
2AE3  FC 42 2B    2B42 'BREAK KEY  EDI output '*BREAK*'
2AE6  EF 40 2B    2B40 'BREAK mark EDI ESC key pressed (first time)
2AE9  0D 69 2B    2B69 'CR  (^M)   EDI function key 'Enter
2AEC  F0 B3 2B    2BB3 'CURSOR up  EDI function key 'CURSOR UP
2AEF  F1 7E 2B    2B7E 'CURSOR downEDI function key 'CURSOR DOWN
2AF2  F2 AA 2B    2BAA 'CURSOR leftEDI function key 'CURSOR LEFT
2AF5  F3 75 2B    2B75 'CURSOR righEDI function key 'CURSOR RIGHT
2AF8  F8 C7 2B    2BC7 'CURSOR SoTXEDI function key 'CURSOR start of text
2AFB  F9 92 2B    2B92 'CURSOR EoTXEDI function key 'CURSOR end of text
2AFE  FA BD 2B    2BBD 'CURSOR SoLNEDI function key 'CURSOR start of line
2B01  FB 89 2B    2B89 'CURSOR EoLNEDI function key 'CURSOR end of line
2B04  F4 A2 2C    2CA2 'COPYCU up  EDI function key 'COPYCURSOR UP
2B07  F5 A7 2C    2CA7 'COPYCU downEDI function key 'COPYCURSOR DOWN
2B0A  F6 9D 2C    2C9D 'COPYCU leftEDI function key 'COPYCURSOR LEFT
2B0D  F7 98 2C    2C98 'COPYCU righEDI function key 'COPYCURSOR RIGHT
2B10  E0 EA 2C    2CEA 'COPY KEY   EDI function key 'COPY
2B13  7F 3D 2C    2C3D 'DEL        EDI function key 'DEL
2B16  10 4A 2C    2C4A 'DLE (^P)   EDI function key 'DLE (CLR)
2B19  E1 F9 2B    2BF9 'INS (^TAB) EDI function key 'INSERT (^TAB)

----- table of cursor functions and 'BEL
      @ 2AD8:
2B1C  04 2B 2B    2B2B 'EOT (^D)   EDI output 'BEL
2B1F  F0 2F 2B    2B2F 'CURSOR up  EDI perform CURSOR UP
2B22  F1 33 2B    2B33 'CURSOR downEDI perform CURSOR DOWN
2B25  F2 3B 2B    2B3B 'CURSOR leftEDI perform CURSOR RIGHT
2B28  F3 37 2B    2B37 'CURSOR righEDI perform CURSOR LEFT

----- EDI output 'BEL
      @ 2B7A  2B84  2BAF  2BB8  2C1A  2C3F  2C45  2C4C  2D26
2B2B  3E 07       ld a,07          'BEL (^G)
2B2D  18 0E       jr 2B3D

----- EDI perform CURSOR UP
      @ 2B1F!
2B2F  3E 0B       ld a,0B          'VT  (^K)
2B31  18 0A       jr 2B3D

----- EDI perform CURSOR DOWN
      @ 2B22!
2B33  3E 0A       ld a,0A          'LF  (^J)
2B35  18 06       jr 2B3D

----- EDI perform CURSOR LEFT
      @ 2B28!
2B37  3E 09       ld a,09          'HT  (^I)
2B39  18 02       jr 2B3D

----- EDI perform CURSOR RIGHT
      @ 2B25!
2B3B  3E 08       ld a,08          'BS  (^H)
2B3D  CD 00 14    call 1400        TXT OUTPUT char or ctl code <a> to VDU

----- EDI ESC key pressed (first time)
      @ 2AE6!
2B40  B7          or a
2B41  C9          ret

----- EDI output '*BREAK*'
      @ 2AE3!
2B42  F5          push af
2B43  CD 49 2B     call 2B49       do it here
2B46  F1          pop af
```

```
2B47  37          scf
2B48  C9          ret

----- do it here
      @ 2B43!
2B49  CD 69 2B    call 2B69        EDI function key 'Enter
2B4C  21 61 2B    ld hl,2B61       '*Break*
2B4F  CD 69 2B    call 2B69        EDI function key 'Enter
2B52  CD 80 11    call 1180        TXT GET CURSOR position (hl), roll count <a>
2B55  25          dec h
2B56  C8          ret z            if already in column 1
2B57  3E 0D       ld a,0D          'CR  (^M)
2B59  CD 00 14    call 1400        TXT OUTPUT char or ctl code <a> to VDU
2B5C  3E 0A       ld a,0A          'LF  (^J)
2B5E  C3 00 14    jp 1400          TXT OUTPUT char or ctl code <a> to VDU

----- '*Break*
2B61  2A 42 72 65 61 6B 2A 00                              '*Break*.

----- EDI function key 'Enter
      @ 2AE9! 2B49! 2B4F!
2B69  F5          push af
2B6A  7E           ld a,(hl)
2B6B  23           inc hl
2B6C  B7           or a
2B6D  C4 A8 2D     call nz,2DA8    EDI write char <a>, handle both cursors
2B70  20 F8        jr nz,2B6A      ...
2B72  F1          pop af
2B73  37          scf
2B74  C9          ret

----- EDI function key 'CURSOR RIGHT
      @ 2AF5!
2B75  16 01       ld d,01          +1
2B77  CD 93 2B    call 2B93        EDI move cursor <d> steps right
2B7A  CA 2B 2B    jp z,2B2B        EDI output 'BEL
2B7D  C9          ret

----- EDI function key 'CURSOR DOWN
      @ 2AEF!
2B7E  CD EB 2B    call 2BEB        EDI get window width <d>; current column <e>
2B81  79          ld a,c
2B82  90          sub b
2B83  BA          cp d
2B84  DA 2B 2B    jp c,2B2B        EDI output 'BEL
2B87  18 0A       jr 2B93          EDI move cursor <d> steps right

----- EDI function key 'CURSOR end of line
      @ 2B01!
2B89  CD EB 2B    call 2BEB        EDI get window width <d>; current column <e>
2B8C  7A          ld a,d           window width
2B8D  93          sub e            is cursor at the end already?
2B8E  C8          ret z            yes, return
2B8F  57          ld d,a
2B90  18 01       jr 2B93          EDI move cursor <d> steps right

----- EDI function key 'CURSOR end of text
      @ 2AFB!
2B92  51          ld d,c

----- EDI move cursor <d> steps right
      @ 2B77! 2B87' 2B90' 2BA5'
2B93  78          ld a,b
2B94  B9          cp c
2B95  C8          ret z
```

```
2B96  D5          push de
2B97  CD 50 2D     call 2D50        EDI move COPYCURSOR right
2B9A  7E           ld a,(hl)
2B9B  D4 A8 2D     call nc,2DA8     EDI write char <a>, handle both cursors
2B9E  04           inc b
2B9F  23           inc hl
2BA0  D4 67 2D     call nc,2D67     EDI write string at cursor pos; update curso
2BA3  D1          pop de
2BA4  15          dec d
2BA5  20 EC       jr nz,2B93        EDI move cursor <d> steps right
2BA7  F6 FF       or FF             clear carry
2BA9  C9          ret

----- EDI function key 'CURSOR LEFT
      @ 2AF2!
2BAA  16 01       ld d,01           +1
2BAC  CD C8 2B    call 2BC8         EDI move cursor <d> steps left
2BAF  CA 2B 2B    jp z,2B2B         EDI output 'BEL
2BB2  C9          ret

----- EDI function key 'CURSOR UP
      @ 2AEC!
2BB3  CD EB 2B    call 2BEB         EDI get window width <d>; current column <e>
2BB6  78          ld a,b
2BB7  BA          cp d
2BB8  DA 2B 2B    jp c,2B2B         EDI output 'BEL
2BBB  18 0B       jr 2BC8           EDI move cursor <d> steps left

----- EDI function key 'CURSOR start of line
      @ 2AFE!
2BBD  CD EB 2B    call 2BEB         EDI get window width <d>; current column <e>
2BC0  7B          ld a,e
2BC1  D6 01       sub 01            column -1
2BC3  C8          ret z             was already column one, return
2BC4  57          ld d,a
2BC5  18 01       jr 2BC8           EDI move cursor <d> steps left

----- EDI function key 'CURSOR start of text
      @ 2AF8!
2BC7  51          ld d,c

----- EDI move cursor <d> steps left
      @ 2BAC! 2BBB' 2BC5' 2BD3'
2BC8  78          ld a,b
2BC9  B7          or a
2BCA  C8          ret z
2BCB  CD 4A 2D    call 2D4A         EDI move COPYCURSOR left
2BCE  30 07       jr nc,2BD7        ...
2BD0  05          dec b
2BD1  2B          dec hl
2BD2  15          dec d
2BD3  20 F3       jr nz,2BC8        EDI move cursor <d> steps left
2BD5  18 11       jr 2BE8

2BD7  78          ld a,b
2BD8  B7          or a
2BD9  28 0A       jr z,2BE5         ...
2BDB  05          dec b
2BDC  2B          dec hl
2BDD  D5          push de
2BDE  CD 29 2D     call 2D29        EDI move COPYCURSOR up
2BE1  D1          pop de
2BE2  15          dec d
2BE3  20 F2       jr nz,2BD7        ...
2BE5  CD 67 2D    call 2D67         EDI write string at cursor pos; update curso
```

```
2BE8  F6 FF        or FF              clear carry
2BEA  C9           ret

----- EDI get window width <d>; current column <e>
      @ 2B7E! 2B89! 2BB3! 2BBD!
2BEB  E5           push hl
2BEC  CD 56 12      call 1256         TXT GET WINDOW size, <hl>=left top, <de>=rig
2BEF  7A            ld a,d
2BF0  94            sub h             <d>=<d>-<h>; <e>=cursor column
2BF1  3C            inc a
2BF2  57            ld d,a
2BF3  CD 80 11      call 1180         TXT GET CURSOR position (hl), roll count <a>
2BF6  5C            ld e,h
2BF7  E1           pop hl
2BF8  C9           ret

----- EDI function key 'INSERT (^TAB)
      @ 2B19!
2BF9  3A DD B8     ld a,(B8DD)        EDI INSERT/OVERWRITE flag
2BFC  2F           cpl
2BFD  32 DD B8     ld (B8DD),a        EDI INSERT/OVERWRITE flag
2C00  C9           ret

----- EDI function key 'DC3 (^S)
      @ 2AE0! 2D23
2C01  B7           or a
2C02  C8           ret z
2C03  5F           ld e,a
2C04  3A DD B8     ld a,(B8DD)        EDI INSERT/OVERWRITE flag
2C07  B7           or a
2C08  28 0D        jr z,2C17          ...
2C0A  78           ld a,b
2C0B  B9           cp c
2C0C  28 09        jr z,2C17          ...
2C0E  73           ld (hl),e
2C0F  7B           ld a,e
2C10  CD A8 2D     call 2DA8          EDI write char <a>, handle both cursors
2C13  23           inc hl
2C14  04           inc b
2C15  B7           or a
2C16  C9           ret

2C17  79           ld a,c
2C18  FE FF        cp FF              'IGNORE
2C1A  CA 2B 2B     jp z,2B2B          EDI output 'BEL
2C1D  AF           xor a
2C1E  32 DC B8     ld (B8DC),a        EDI cursor on flag
2C21  7B           ld a,e
2C22  CD A8 2D     call 2DA8          EDI write char <a>, handle both cursors
2C25  0C           inc c
2C26  E5           push hl
2C27  7E            ld a,(hl)
2C28  73            ld (hl),e
2C29  5F            ld e,a
2C2A  23            inc hl
2C2B  B7            or a
2C2C  20 F9         jr nz,2C27        ...
2C2E  77            ld (hl),a
2C2F  E1           pop hl
2C30  04           inc b
2C31  23           inc hl
2C32  CD 67 2D     call 2D67          EDI write string at cursor pos; update curso
2C35  3A DC B8     ld a,(B8DC)        EDI cursor on flag
2C38  B7           or a
2C39  C4 29 2D     call nz,2D29       EDI move COPYCURSOR up
```

```
2C3C  C9           ret

----- EDI function key 'DEL
      @ 2B13!
2C3D  78           ld a,b            <b>=cursor pos within string (0 ... n)
2C3E  B7           or a              already on first position?
2C3F  CA 2B 2B     jp z,2B2B         EDI output 'BEL
2C42  CD 4A 2D     call 2D4A         EDI move COPYCURSOR left
2C45  D2 2B 2B     jp nc,2B2B        EDI output 'BEL
2C48  05           dec b
2C49  2B           dec hl

----- EDI function key 'DLE (CLR)
      @ 2B16
2C4A  78           ld a,b
2C4B  B9           cp c              <c>=len of string
2C4C  CA 2B 2B     jp z,2B2B         EDI output 'BEL
2C4F  E5           push hl
2C50  23            inc hl
2C51  7E            ld a,(hl)
2C52  2B            dec hl
2C53  77            ld (hl),a
2C54  23            inc hl
2C55  B7            or a
2C56  20 F8         jr nz,2C50       shift rest of string towards the cursor
2C58  2B            dec hl           delete last char on the screen with a
2C59  36 20         ld (hl),20       'SPACE
2C5B  32 DC B8      ld (B8DC),a      EDI cursor on flag
2C5E  E3            ex (sp),hl
2C5F  CD 67 2D      call 2D67        EDI write string at cursor pos; update curso
2C62  E3            ex (sp),hl
2C63  36 00         ld (hl),00       mark end of string
2C65  E1           pop hl
2C66  0D           dec c
2C67  3A DC B8     ld a,(B8DC)       EDI cursor on flag
2C6A  B7           or a
2C6B  C4 2D 2D     call nz,2D2D      EDI move COPYCURSOR down
2C6E  C9           ret

----- EDI reset COPYCURSOR to zero
      @ 2AA8!
2C6F  21 00 00     ld hl,0000
2C72  22 DE B8     ld (B8DE),hl      EDI COPYCURSOR position column/row
2C75  C9           ret

----- compare COPYCURSOR with CURSOR; =carry
      @ 2DB6! 2DE0!
2C76  ED 5B DE B8 ld de,(B8DE)       EDI COPYCURSOR position column/row
2C7A  7C           ld a,h
2C7B  AA           xor d
2C7C  C0           ret nz
2C7D  7D           ld a,l
2C7E  AB           xor e
2C7F  C0           ret nz
2C80  37           scf
2C81  C9           ret

----- add roll count to COPYCURSOR, validate, reset if illegal
      @ 2DC7! 2DED!
2C82  4F           ld c,a            roll count
2C83  2A DE B8     ld hl,(B8DE)      EDI COPYCURSOR position column/row
2C86  7C           ld a,h
2C87  B5           or l
2C88  C8           ret z             if not present
2C89  7D           ld a,l
```

```
2C8A  81          add a,c           add roll count to vertical position
2C8B  6F          ld l,a

----- validate COPYCURSOR, reset if illegal
      @ 2D40!
2C8C  CD CE 11    call 11CE         TXT VALIDATE cursor position <hl> column/row
2C8F  38 03       jr c,2C94         window would not roll
2C91  21 00 00    ld hl,0000        mark COPYCURSOR not present
2C94  22 DE B8    ld (B8DE),hl      EDI COPYCURSOR position column/row
2C97  C9          ret

----- EDI function key 'COPYCURSOR RIGHT
      @2B0D!
2C98  11 00 01    ld de,0100        <d>=+1 horizontal move; <e> unchanged
2C9B  18 0D       jr 2CAA

----- EDI function key 'COPYCURSOR LEFT
      @2B0A!
2C9D  11 00 FF    ld de,FF00        <d>=-1 horizontal move
2CA0  18 08       jr 2CAA

----- EDI function key 'COPYCURSOR UP
      @ 2B04!
2CA2  11 FF 00    ld de,00FF        <d> unchanged, <e>=-1 vertical move
2CA5  18 03       jr 2CAA

----- EDI function key 'COPYCURSOR DOWN
      @ 2B07!
2CA7  11 01 00    ld de,0001        vertical move down
2CAA  C5          push bc
2CAB  E5           push hl
2CAC  2A DE B8      ld hl,(B8DE)    EDI COPYCURSOR position column/row
2CAF  7C            ld a,h
2CB0  B5            or l
2CB1  CC 80 11      call z,1180     TXT GET CURSOR position (hl), roll count <a>
2CB4  7C            ld a,h
2CB5  82            add a,d
2CB6  67            ld h,a
2CB7  7D            ld a,l
2CB8  83            add a,e
2CB9  6F            ld l,a
2CBA  CD CE 11      call 11CE       TXT VALIDATE cursor position <hl> column/row
2CBD  30 0B         jr nc,2CCA      if printing would roll the window
2CBF  E5            push hl
2CC0  CD D2 2C       call 2CD2      remove old cursor, place at new location
2CC3  E1            pop hl
2CC4  22 DE B8      ld (B8DE),hl    EDI COPYCURSOR position column/row
2CC7  CD CD 2C      call 2CCD       remove old cursor, place at new location
2CCA  E1           pop hl
2CCB  C1          pop bc
2CCC  C9          ret

----- remove old cursor, place at new location
      @ 2CC7! 2D1D! 2D43! 2DD1!
2CCD  11 68 12    ld de,1268        TXT PLACE/REMOVE CURSOR on screen
2CD0  18 03       jr 2CD5

----- remove old cursor, place at new location
      @ 2ABC! 2CC0! 2D32! 2DBA!
2CD2  11 68 12    ld de,1268        TXT PLACE/REMOVE CURSOR on screen
2CD5  2A DE B8    ld hl,(B8DE)      EDI COPYCURSOR position column/row
2CD8  7C          ld a,h
2CD9  B5          or l
2CDA  C8          ret z             it's not there
2CDB  E5          push hl
```

```
2CDC  CD 80 11     call 1180        TXT GET CURSOR position (hl), roll count <a>
2CDF  E3           ex (sp),hl
2CE0  CD 74 11     call 1174        TXT SET CURSOR, <hl>=column/row
2CE3  CD 16 00     call 0016        jp(de); always remove old, place new cursor
2CE6  E1          pop hl
2CE7  C3 74 11    jp 1174           TXT SET CURSOR, <hl>=column/row

----- EDI function key 'COPY
      @ 2B10!
2CEA  C5          push bc
2CEB  E5           push hl
2CEC  CD 80 11      call 1180       TXT GET CURSOR position (hl), roll count <a>
2CEF  EB            ex de,hl
2CF0  2A DE B8      ld hl,(B8DE)    EDI COPYCURSOR position column/row
2CF3  7C            ld a,h
2CF4  B5            or l
2CF5  20 0C         jr nz,2D03      there is a COPYCURSOR
2CF7  78            ld a,b
2CF8  B1            or c
2CF9  20 26         jr nz,2D21      ...
2CFB  CD 80 11      call 1180       TXT GET CURSOR position (hl), roll count <a>
2CFE  22 DE B8      ld (B8DE),hl    EDI COPYCURSOR position column/row
2D01  18 06         jr 2D09

2D03  CD 74 11      call 1174       TXT SET CURSOR, <hl>=column/row
2D06  CD 68 12      call 1268       TXT PLACE/REMOVE CURSOR on screen
2D09  CD AB 13      call 13AB       TXT READ char from screen <hl>=col/row, =<a>
2D0C  F5            push af
2D0D  EB             ex de,hl
2D0E  CD 74 11       call 1174      TXT SET CURSOR, <hl>=column/row
2D11  2A DE B8       ld hl,(B8DE)   EDI COPYCURSOR position column/row
2D14  24             inc h
2D15  CD CE 11       call 11CE      TXT VALIDATE cursor position <hl> column/row
2D18  30 03          jr nc,2D1D     if printing would roll the window
2D1A  22 DE B8       ld (B8DE),hl   EDI COPYCURSOR position column/row
2D1D  CD CD 2C       call 2CCD      remove old cursor, place at new location
2D20  F1            pop af
2D21  E1           pop hl
2D22  C1          pop bc
2D23  DA 01 2C    jp c,2C01         EDI function key 'DC3 (^S)
2D26  C3 2B 2B    jp 2B2B           EDI output 'BEL

----- EDI move COPYCURSOR up
      @ 2BDE! 2C39!
2D29  16 01       ld d,01           +1
2D2B  18 02       jr 2D2F

----- EDI move COPYCURSOR down
      @ 2C6B!
2D2D  16 FF       ld d,FF           -1
2D2F  C5          push bc
2D30  E5           push hl
2D31  D5            push de
2D32  CD D2 2C       call 2CD2      remove old cursor, place at new location
2D35  D1            pop de
2D36  2A DE B8      ld hl,(B8DE)    EDI COPYCURSOR position column/row
2D39  7C            ld a,h
2D3A  B5            or l
2D3B  28 09         jr z,2D46       COPYCURSOR not set
2D3D  7C            ld a,h
2D3E  82            add a,d
2D3F  67            ld h,a
2D40  CD 8C 2C      call 2C8C       validate COPYCURSOR, reset if illegal
2D43  CD CD 2C      call 2CCD       remove old cursor, place at new location
2D46  E1           pop hl
```

```
2D47  C1          pop bc
2D48  B7          or a
2D49  C9          ret

----- EDI move COPYCURSOR left
      @ 2BCB! 2C42!
2D4A  D5          push de
2D4B  11 08 FF     ld de,FF08      <d>=-1; <e>='BS
2D4E  18 04        jr 2D54

----- EDI move COPYCURSOR right
      @ 2B97!
2D50  D5          push de
2D51  11 09 01     ld de,0109      <d>=+1; <e>='HT
2D54  C5           push bc
2D55  E5            push hl
2D56  CD 80 11       call 1180     TXT GET CURSOR position (hl), roll count <a>
2D59  7A             ld a,d        = +1/-1
2D5A  84             add a,h       add to column
2D5B  67             ld h,a
2D5C  CD CE 11       call 11CE     TXT VALIDATE cursor position <hl> column/row
2D5F  7B             ld a,e        = 'HT/'BS
2D60  DC 00 14       call c,1400   TXT OUTPUT char or ctl code <a> to VDU
2D63  E1            pop hl
2D64  C1           pop bc
2D65  D1          pop de
2D66  C9          ret

----- EDI write string at cursor pos; update cursor
      @ 2AAC! 2BA0! 2BE5! 2C32! 2C5F!
2D67  C5          push bc
2D68  E5           push hl
2D69  EB            ex de,hl       move string address to <de>
2D6A  CD 80 11      call 1180      TXT GET CURSOR position (hl), roll count <a>
2D6D  4F            ld c,a         roll count
2D6E  EB            ex de,hl       (de)=cursor pos now
2D6F  7E            ld a,(hl)      get a char from textstring
2D70  23            inc hl         prepare for next
2D71  B7            or a           test this char
2D72  C4 85 2D      call nz,2D85   ...
2D75  20 F8         jr nz,2D6F     string not terminated, get next char
2D77  CD 80 11      call 1180      TXT GET CURSOR position (hl), roll count <a>
2D7A  91            sub c          roll count - old roll count
2D7B  EB            ex de,hl
2D7C  85            add a,l
2D7D  6F            ld l,a
2D7E  CD 74 11      call 1174      TXT SET CURSOR, <hl>=column/row
2D81  E1           pop hl
2D82  C1          pop bc
2D83  B7          or a
2D84  C9          ret

      @ 2D72!
2D85  F5          push af
2D86  C5           push bc
2D87  D5            push de
2D88  E5             push hl
2D89  47              ld b,a       save char in <b>
2D8A  CD 80 11        call 1180    TXT GET CURSOR position (hl), roll count <a>
2D8D  91              sub c
2D8E  83              add a,e
2D8F  5F              ld e,a       <e>=<e>+<roll count>-<old roll count>
2D90  48              ld c,b       this is the char
2D91  CD CE 11        call 11CE    TXT VALIDATE cursor position <hl> column/row
2D94  38 05           jr c,2D9B    the window would not roll
```

```
2D96  78                ld a,b
2D97  87                add a,a
2D98  3C                inc a
2D99  83                add a,e
2D9A  5F                ld e,a
2D9B  EB                ex de,hl      <hl> is cursor pos now
2D9C  CD CE 11          call 11CE     TXT VALIDATE cursor position <hl> column/row
2D9F  79                ld a,c        =char;    then if window would not roll:
2DA0  DC A8 2D          call c,2DA8   EDI write char <a>, handle both cursors
2DA3  E1               pop hl
2DA4  D1              pop de
2DA5  C1             pop bc
2DA6  F1            pop af
2DA7  C9            ret

----- EDI write char <a>, handle both cursors
      @ 2B6D! 2B9B! 2C10! 2C22! 2DA0!
2DA8  F5            push af
2DA9  C5             push bc
2DAA  D5              push de
2DAB  E5               push hl
2DAC  47                ld b,a        =char
2DAD  CD 80 11          call 1180     TXT GET CURSOR position (hl), roll count <a>
2DB0  4F                ld c,a        roll count
2DB1  C5                push bc
2DB2  CD CE 11           call 11CE    TXT VALIDATE cursor position <hl> column/row
2DB5  C1                pop bc
2DB6  DC 76 2C          call c,2C76   compare COPYCURSOR with CURSOR; =carry
2DB9  F5                push af
2DBA  DC D2 2C           call c,2CD2 remove old cursor, place at new location
2DBD  78                 ld a,b       =char
2DBE  C5                 push bc
2DBF  CD 34 13            call 1334   TXT WRITE char <a> to screen
2DC2  C1                 pop bc
2DC3  CD 80 11           call 1180    TXT GET CURSOR position (hl), roll count <a>
2DC6  91                 sub c
2DC7  C4 82 2C           call nz,2C82add roll count to COPYCURSOR, validate, rese
2DCA  F1                pop af
2DCB  30 07             jr nc,2DD4    if COPYCURSOR <> CURSOR
2DCD  9F                sbc a,a       =0; reset
2DCE  32 DC B8          ld (B8DC),a   EDI cursor on flag
2DD1  CD CD 2C          call 2CCD     remove old cursor, place at new location
2DD4  E1               pop hl
2DD5  D1              pop de
2DD6  C1             pop bc
2DD7  F1            pop af
2DD8  C9            ret

----- EDI key-input a char at cursor pos
      @ 2AB1!
2DD9  CD 80 11      call 1180         TXT GET CURSOR position (hl), roll count <a>
2DDC  4F            ld c,a
2DDD  CD CE 11      call 11CE         TXT VALIDATE cursor position <hl> column/row
2DE0  CD 76 2C      call 2C76         compare COPYCURSOR with CURSOR; =carry
2DE3  DA 3C 1A      jp c,1A3C         KM WAIT CHAR from keyboard =<a>
2DE6  CD 79 12      call 1279         TXT CURSOR ON
2DE9  CD 80 11      call 1180         TXT GET CURSOR position (hl), roll count <a>
2DEC  91            sub c
2DED  C4 82 2C      call nz,2C82      add roll count to COPYCURSOR, validate, rese
2DF0  CD 3C 1A      call 1A3C         KM WAIT CHAR from keyboard =<a>
2DF3  C3 81 12      jp 1281           TXT CURSOR OFF
```

```
----- EDI find key token, address = (hl)
      @ 2ADB!
2DF6  F5          push af
2DF7  C5           push bc
2DF8  46            ld b,(hl)
2DF9  23            inc hl
2DFA  E5            push hl
2DFB  23             inc hl
2DFC  23             inc hl
2DFD  BE             cp (hl)
2DFE  23             inc hl
2DFF  28 04          jr z,2E05     ...
2E01  05             dec b
2E02  20 F7          jr nz,2DFB    ...
2E04  E3             ex (sp),hl
2E05  F1            pop af
2E06  7E            ld a,(hl)
2E07  23            inc hl
2E08  66            ld h,(hl)
2E09  6F            ld l,a
2E0A  C1           pop bc
2E0B  F1          pop af
2E0C  C9          ret

2E0D  C7 C7 C7 C7 C7 C7 C7 C7  C7 C7 C7
```

```
----- copy 5 bytes,(de)>(hl); ld a,(hl-1)
      @ 31A6  32B4! 3313  331A  3321! 332C! BD3D!
2E18  E5           push hl
2E19  D5            push de
2E1A  C5             push bc
2E1B  EB              ex de,hl
2E1C  01 05 00        ld bc,0005     count of 5 bytes to copy
2E1F  ED B0           ldir
2E21  EB              ex de,hl
2E22  2B              dec hl
2E23  7E              ld a,(hl)
2E24  C1             pop bc
2E25  D1            pop de
2E26  E1           pop hl
2E27  37           scf
2E28  C9           ret

----- REAL ARITH, CREAL <hl> to (de)
      @ 3075! 32ED! BD40!
2E29  D5           push de
2E2A  C5            push bc
2E2B  F6 7F          or 7F           mask for sign
2E2D  47             ld b,a          save sign
2E2E  AF             xor a           =0
2E2F  12             ld (de),a
2E30  13             inc de
2E31  12             ld (de),a
2E32  13             inc de
2E33  0E 90          ld c,90         =144.
2E35  7C             ld a,h          hi byte zero?
2E36  B7             or a
2E37  20 08          jr nz,2E41      no, it's not
2E39  4F             ld c,a
2E3A  65             ld h,l          get lo byte
2E3B  6F             ld l,a          clear <l>
2E3C  B4             or h            lo byte zero too?
2E3D  28 0D          jr z,2E4C       yes, it's a zero value
2E3F  0E 88          ld c,88         set exponent for int mant (+128.)
2E41  FA 4B 2E       jp m,2E4B       hi byte was negative
2E44  29             add hl,hl       shift mantissa left
2E45  0D             dec c           decrement exponent
2E46  B4             or h            bit 7 set?
2E47  F2 44 2E       jp p,2E44       no, shift again
2E4A  7C             ld a,h
2E4B  A0             and b           set sign bit, if <integer> was negative
2E4C  EB             ex de,hl
2E4D  73             ld (hl),e       store result into real number bytes
2E4E  23             inc hl
2E4F  77             ld (hl),a
2E50  23             inc hl
2E51  71             ld (hl),c
2E52  C1            pop bc
2E53  E1           pop hl
2E54  C9           ret

----- REAL ARITH, CREAL (hl) 4 byte integer to real
      @ BD43!
2E55  C5           push bc
2E56  01 00 A0      ld bc,A000       ...
2E59  CD 60 2E      call 2E60        ...
2E5C  C1           pop bc
2E5D  C9           ret
```

```
----- REAL ARITH ??
      @ BD94!
2E5E  06 A8         ld b,A8            =168.

      @ 2E59!
2E60  D5            push de
2E61  CD A1 36       call 36A1         ix=hl; set exp <b>; get REAL <l><h><e><d>
2E64  D1            pop de
2E65  C9            ret

----- REAL ARITH, CINT; <hl>=int(hl); <a>=sign
      @ 32E1! BD46!
2E66  E5            push hl
2E67  DD E1         pop ix
2E69  AF            xor a
2E6A  DD 96 04      sub (ix+04)        complement exponent
2E6D  28 1B         jr z,2E8A          result is zero
2E6F  C6 90         add a,90           test for overflow
2E71  D0            ret nc
2E72  D5            push de
2E73  C5             push bc
2E74  C6 10           add a,10         =16.
2E76  CD 3D 36        call 363D        ...
2E79  CB 21           sla c
2E7B  ED 5A           adc hl,de
2E7D  28 08           jr z,2E87        ...
2E7F  DD 7E 03        ld a,(ix+03)
2E82  B7              or a
2E83  3F              ccf
2E84  C1             pop bc
2E85  D1            pop de
2E86  C9            ret

2E87  9F              sbc a,a
2E88  18 F9           jr 2E83          ...

----- result is zero
2E8A  6F            ld l,a
2E8B  67            ld h,a
2E8C  37            scf
2E8D  C9            ret

----- REAL ARITH ??
      @ 2F01! BD49!
2E8E  CD A1 2E      call 2EA1          REAL ARITH, FIX (hl)
2E91  D0            ret nc
2E92  F0            ret p
2E93  E5            push hl
2E94  79             ld a,c
2E95  34             inc (hl)
2E96  20 06          jr nz,2E9E        ...
2E98  23             inc hl
2E99  3D             dec a
2E9A  20 F9          jr nz,2E95        ...
2E9C  34             inc (hl)
2E9D  0C             inc c
2E9E  E1            pop hl
2E9F  37            scf
2EA0  C9            ret

----- REAL ARITH, FIX (hl)
      @ 2E8E! 2EAC! 3186! BD4C!
2EA1  E5            push hl
2EA2  D5             push de
2EA3  E5              push hl
```

```
2EA4  DD E1        pop ix
2EA6  CD 04 36     call 3604        ...
2EA9  D1           pop de
2EAA  E1           pop hl
2EAB  C9           ret

----- REAL ARITH, INT (hl)
      @ BD4F!
2EAC  CD A1 2E     call 2EA1        REAL ARITH, FIX (hl)
2EAF  D0           ret nc
2EB0  C8           ret z
2EB1  CB 78        bit 7,b
2EB3  C8           ret z
2EB4  18 DD        jr 2E93          ...

----- REAL ARITH ??
      @ BD52!
2EB6  CD E8 35     call 35E8        REAL ARITH, SGN (hl); <a>=FF,00,01
2EB9  47           ld b,a
2EBA  28 52        jr z,2F0E        ...
2EBC  FC FB 35     call m,35FB      REAL ARITH, COMPLEMENT SIGN (ix)
2EBF  E5           push hl
2EC0  DD 7E 04     ld a,(ix+04)
2EC3  D6 80        sub 80           =128.
2EC5  5F           ld e,a
2EC6  9F           sbc a,a
2EC7  57           ld d,a
2EC8  6B           ld l,e
2EC9  62           ld h,d
2ECA  29           add hl,hl
2ECB  29           add hl,hl
2ECC  29           add hl,hl
2ECD  19           add hl,de
2ECE  29           add hl,hl
2ECF  19           add hl,de
2ED0  29           add hl,hl
2ED1  29           add hl,hl
2ED2  19           add hl,de
2ED3  7C           ld a,h
2ED4  D6 09        sub 09           =9.
2ED6  5F           ld e,a
2ED7  E1           pop hl
2ED8  C5           push bc
2ED9  D5           push de
2EDA  C4 1F 2F     call nz,2F1F     ...
2EDD  FD 21 13 2F  ld iy,2F13       constant 3124999.98
2EE1  CD A0 35     call 35A0        COMPARE (ix),(iy); <a>=FF,00,01
2EE4  28 1B        jr z,2F01        ...
2EE6  30 08        jr nc,2EF0       ...
2EE8  CD 12 34     call 3412        MULTIPLY by 10., (hl)=(hl)*10.
2EEB  D1           pop de
2EEC  1D           dec e
2EED  D5           push de
2EEE  18 ED        jr 2EDD          ...

2EF0  FD 21 18 2F  ld iy,2F18       constant 1 E+9
2EF4  CD A0 35     call 35A0        COMPARE (ix),(iy); <a>=FF,00,01
2EF7  38 08        jr c,2F01        ...
2EF9  CD 9B 34     call 349B        DEVIDE by 10., (hl)=(hl)/10.
2EFC  D1           pop de
2EFD  1C           inc e
2EFE  D5           push de
2EFF  18 EF        jr 2EF0          ...
```

```
2F01  CD 8E 2E      call 2E8E        REAL ARITH ??
2F04  79            ld a,c
2F05  D1            pop de
2F06  C1            pop bc
2F07  4F            ld c,a
2F08  3D            dec a
2F09  85            add a,l
2F0A  6F            ld l,a
2F0B  D0            ret nc
2F0C  24            inc h
2F0D  C9            ret

2F0E  5F            ld e,a
2F0F  77            ld (hl),a
2F10  0E 01         ld c,01          =1.
2F12  C9            ret

----- constant 3124999.98
2F13  F0 1F BC 3E 96

----- constant 1 E+9
2F18  FE 27 6B 6E 9E

----- REAL ARITH ??
      @ BD55!
2F1D  2F            cpl
2F1E  3C            inc a

      @ 2EDA!
2F1F  B7            or a
2F20  37            scf
2F21  C8            ret z
2F22  4F            ld c,a
2F23  F2 28 2F      jp p,2F28        ...
2F26  2F            cpl
2F27  3C            inc a
2F28  CD 3E 2F      call 2F3E        ...
2F2B  28 09         jr z,2F36        ...
2F2D  C5            push bc
2F2E  F5            push af
2F2F  CD 36 2F      call 2F36        ...
2F32  F1            pop af
2F33  C1            pop bc
2F34  18 F2         jr 2F28          ...

      @ 2F2B' 2F2F!
2F36  79            ld a,c
2F37  B7            or a
2F38  F2 9E 34      jp p,349E        REAL ARITH DVD; (hl)=(hl)/(de)
2F3B  C3 15 34      jp 3415          REAL ARITH, MULT, (hl)=(hl)*(de)

      @ 2F28!
2F3E  11 8F 2F      ld de,2F8F       ...
2F41  D6 0D         sub 0D           =13.
2F43  D0            ret nc
2F44  C6 0C         add a,0C         =12.
2F46  5F            ld e,a
2F47  87            add a,a          *2
2F48  87            add a,a          *4
2F49  83            add a,e          +1
2F4A  C6 53         add a,53
2F4C  5F            ld e,a
2F4D  CE 2F         adc a,2F         <de>=<a>*5+2F53
2F4F  93            sub e
2F50  57            ld d,a
```

```
2F51  AF          xor a            =0
2F52  C9          ret

----- table of constants 10. ... 1E+13
      @ 2F4D: 3412: 349B:  (2F8F) 2F3E:
2F53  00 00 00 20 84               constant 10.
2F58  00 00 00 48 87               constant 100.
2F5D  00 00 00 7A 8A               constant 1000.
2F62  00 00 40 1C 8E               constant 10000.
2F67  00 00 50 43 91               constant 100000.
2F6C  00 00 24 74 94               constant 1000000.
2F71  00 80 96 18 98               constant 10000000.
2F76  00 20 BC 3E 9B               constant 100000000.
2F7B  00 28 6B 6E 9E               constant 1 E+9
2F80  00 F9 02 15 A2               constant 1 E+10
2F85  40 B7 43 3A A5               constant 1 E+11
2F8A  10 A5 D4 68 A8               constant 1 E+12
2F8F  2A E7 84 11 AC               constant 1 E+13

----- REAL ARITH, set initial RANDOM NUMBER
      @ 2FA2! BD97!
2F94  21 65 89    ld hl,8965
2F97  22 E6 B8    ld (B8E6),hl     RANDOM NUMBER byte 2,3
2F9A  21 07 6C    ld hl,6C07
2F9D  22 E4 B8    ld (B8E4),hl     RANDOM NUMBER byte 0,1
2FA0  C9          ret

----- REAL ARITH, seed RANDOM NUMBER
      @ BD9A!
2FA1  EB          ex de,hl
2FA2  CD 94 2F    call 2F94        REAL ARITH, set initial RANDOM NUMBER
2FA5  EB          ex de,hl
2FA6  CD E8 35    call 35E8        REAL ARITH, SGN (hl); <a>=FF,00,01
2FA9  C8          ret z            RND(0) always returns the initial value
2FAA  11 E4 B8    ld de,B8E4       RANDOM NUMBER byte 0,1
2FAD  06 04       ld b,04          =4.
2FAF  1A          ld a,(de)
2FB0  AE          xor (hl)
2FB1  12          ld (de),a
2FB2  13          inc de
2FB3  23          inc hl
2FB4  10 F9       djnz 2FAF        repeat 4 times
2FB6  C9          ret

----- REAL ARITH, RANDOMIZE
      @ BD9D!
2FB7  E5          push hl
2FB8  2A E6 B8     ld hl,(B8E6)    RANDOM NUMBER byte 2,3
2FBB  01 07 6C     ld bc,6C07      initial val
2FBE  CD FA 2F     call 2FFA       produce a random number
2FC1  E5           push hl
2FC2  2A E4 B8      ld hl,(B8E4)   RANDOM NUMBER byte 0,1
2FC5  01 65 89      ld bc,8965     initial val
2FC8  CD FA 2F      call 2FFA      produce a random number
2FCB  D5            push de
2FCC  E5             push hl
2FCD  2A E6 B8        ld hl,(B8E6) RANDOM NUMBER byte 2,3
2FD0  CD FA 2F        call 2FFA    produce a random number
2FD3  E3              ex (sp),hl
2FD4  09              add hl,bc
2FD5  22 E4 B8        ld (B8E4),hl RANDOM NUMBER byte 0,1
2FD8  E1             pop hl
2FD9  01 07 6C       ld bc,6C07    initial val
2FDC  ED 4A          adc hl,bc
2FDE  C1            pop bc
```

```
2FDF  09            add hl,bc
2FE0  C1           pop bc
2FE1  09           add hl,bc
2FE2  22 E6 B8     ld (B8E6),hl     RANDOM NUMBER byte 2,3
2FE5  E1          pop hl

----- REAL ARITH, get last RANDOM NUMBER (hl)
      @ BDA0!
2FE6  E5          push hl
2FE7  DD E1       pop ix
2FE9  2A E4 B8    ld hl,(B8E4)     RANDOM NUMBER byte 0,1
2FEC  ED 5B E6 B8 ld de,(B8E6)     RANDOM NUMBER byte 2,3
2FF0  01 00 00    ld bc,0000
2FF3  DD 36 04 80 ld (ix+04),80    =128.
2FF7  C3 B1 36    jp 36B1          ...

----- produce a random number
      @ 2FBE! 2FC8! 2FD0!
2FFA  EB          ex de,hl
2FFB  21 00 00    ld hl,0000
2FFE  3E 11       ld a,11          =17.
3000  3D          dec a
3001  C8          ret z
3002  29          add hl,hl
3003  CB 13       rl e
3005  CB 12       rl d
3007  30 F7       jr nc,3000       ...
3009  09          add hl,bc
300A  30 F4       jr nc,3000       ...
300C  13          inc de
300D  18 F1       jr 3000          loop back

----- REAL ARITH, LOG10 (hl)
      @ BD82!
300F  11 8B 30    ld de,308B       constant 0.301029996
3012  18 03       jr 3017

----- REAL ARITH, LOG (hl)
      @ 312D! BD7F!
3014  11 86 30    ld de,3086       constant 0.693147181
3017  CD E8 35    call 35E8        REAL ARITH, SGN (hl); <a>=FF,00,01
301A  3D          dec a
301B  FE 01       cp 01            =1.
301D  D0          ret nc
301E  D5          push de
301F  CD 6C 35     call 356C       test expo (hl); z=zero, c=neg; ix=hl
3022  F5           push af
3023  DD 36 04 80   ld (ix+04),80  =128.
3027  11 81 30      ld de,3081     constant 0.707106781
302A  CD 9A 35      call 359A      REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01
302D  30 06         jr nc,3035     ...
302F  DD 34 04      inc (ix+04)
3032  F1           pop af
3033  3D           dec a
3034  F5           push af
3035  CD 16 33      call 3316      REAL ARITH, copy (hl) to FAC3
3038  D5            push de
3039  11 32 33       ld de,3332    constant 1.
303C  CD 3F 33       call 333F     REAL ARITH, ADD, (hl)=(hl)+(de)
303F  EB             ex de,hl
3040  E1            pop hl
3041  D5            push de
3042  11 32 33       ld de,3332    constant 1.
3045  CD 37 33       call 3337     REAL ARITH, SUB, (hl)=(hl)-(de)
3048  D1            pop de
```

```
3049  CD 9E 34        call 349E       REAL ARITH DVD; (hl)=(hl)/(de)
304C  CD A9 32        call 32A9       ...
304F  04              =4.
3050  4C 4B 57 5E 7F                  constant 0.434259751
3055  0D 08 9B 13 80                  constant 0.576584342
305A  23 93 38 76 80                  constant 0.961800762
305F  20 3B AA 38 82                  constant 2.88539007
3064  D5              push de
3065  CD 15 34         call 3415      REAL ARITH, MULT, (hl)=(hl)*(de)
3068  D1              pop de
3069  E3              ex (sp),hl
306A  7C              ld a,h
306B  B7              or a
306C  F2 71 30        jp p,3071       ...
306F  2F              cpl
3070  3C              inc a
3071  6F              ld l,a
3072  7C              ld a,h
3073  26 00           ld h,00         =0.
3075  CD 29 2E        call 2E29       REAL ARITH, CREAL <hl> to (de)
3078  EB              ex de,hl
3079  E1             pop hl
307A  CD 3F 33       call 333F        REAL ARITH, ADD, (hl)=(hl)+(de)
307D  D1            pop de
307E  C3 15 34      jp 3415           REAL ARITH, MULT, (hl)=(hl)*(de)

----- constant 0.707106781
3081  34 F3 04 35 80

----- constant 0.693147181
3086  F8 17 72 31 80

----- constant 0.301029996
308B  85 9A 20 1A 7F

----- REAL ARITH, get EXP
      @ 3134! BD85!
3090  06 E1         ld b,E1           =225.
3092  CD 07 33      call 3307         test exponent (hl), cp with <b>; set p,z,c
3095  D2 28 33      jp nc,3328        REAL ARITH, copy constant 1. to (hl)
3098  11 00 31      ld de,3100        constant 88.0296919
309B  CD 9A 35      call 359A         REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01
309E  F2 EC 36      jp p,36EC         ...
30A1  11 05 31      ld de,3105        constant -88.7228391
30A4  CD 9A 35      call 359A         REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01
30A7  FA E6 36      jp m,36E6         set exp REAL(ix+4) to 0; set carry; hl=ix
30AA  11 FB 30      ld de,30FB        constant 1.44269504
30AD  CD D4 32      call 32D4         ...
30B0  7B            ld a,e
30B1  F2 B6 30      jp p,30B6         ...
30B4  ED 44         neg
30B6  F5            push af
30B7  CD 1D 33       call 331D        REAL ARITH, copy (hl) to FAC2; mult by (de)
30BA  CD 0F 33       call 330F        REAL ARITH, copy (hl) to FAC1
30BD  D5             push de
30BE  CD AC 32        call 32AC       ...
30C1  03              =3.
30C2  F4 32 EB 0F 73                  constant 6.86258 E-05
30C7  08 B8 D5 52 7B                  constant 2.57367 E-02
30CC  00 00 00 00 80                  constant 0.5
30D1  E3              ex (sp),hl
30D2  CD AC 32        call 32AC       ...
30D5  02              =2.
30D6  09 60 DE 01 78                  constant 1.98164 E-03
30DB  F8 17 72 31 7E                  constant 0.173286795
```

```
30E0  CD 15 34     call 3415        REAL ARITH, MULT, (hl)=(hl)*(de)
30E3  D1          pop de
30E4  E5          push hl
30E5  EB           ex de,hl
30E6  CD 37 33     call 3337        REAL ARITH, SUB, (hl)=(hl)-(de)
30E9  EB           ex de,hl
30EA  E1          pop hl
30EB  CD 9E 34    call 349E         REAL ARITH DVD; (hl)=(hl)/(de)
30EE  11 CC 30    ld de,30CC        constant 0.5
30F1  CD 3F 33    call 333F         REAL ARITH, ADD, (hl)=(hl)+(de)
30F4  DD 34 04    inc (ix+04)
30F7  F1         pop af
30F8  C3 7B 35   jp 357B            ...

30FB  29         add hl,hl          constant 1.44269504
30FC  3B         dec sp
30FD  AA         xor d
30FE  38 81      jr c,3081          constant 0.707106781
3100  C7         rst 0              constant 88.0296919

3101  33         inc sp
3102  0F         rrca
3103  30 87      jr nc,308C         ...
3105  F8         ret m              constant -88.7228391
3106  17         rla
3107  72         ld (hl),d
3108  B1         or c
3109  87         add a,a

----- REAL ARITH, SQR (hl) ^ 0.5
      @ BD79!
310A  11 CC 30   ld de,30CC         constant 0.5

----- REAL ARITH, EXP; (hl)=(hl)^(de)
      @ BD7C!
310D  EB         ex de,hl
310E  CD E8 35   call 35E8          REAL ARITH, SGN (hl); <a>=FF,00,01
3111  EB         ex de,hl
3112  CA 28 33   jp z,3328          REAL ARITH, copy constant 1. to (hl)
3115  F5         push af
3116  CD E8 35    call 35E8         REAL ARITH, SGN (hl); <a>=FF,00,01
3119  28 25       jr z,3140         ...
311B  47          ld b,a
311C  FC FB 35    call m,35FB       REAL ARITH, COMPLEMENT SIGN (ix)
311F  E5          push hl
3120  CD 82 31     call 3182        ...
3123  E1          pop hl
3124  38 25       jr c,314B         ...
3126  E3          ex (sp),hl
3127  E1         pop hl
3128  FA 48 31   jp m,3148          ...
312B  C5         push bc
312C  D5          push de
312D  CD 14 30     call 3014        REAL ARITH, LOG (hl)
3130  D1          pop de
3131  DC 15 34    call c,3415       REAL ARITH, MULT, (hl)=(hl)*(de)
3134  DC 90 30    call c,3090       REAL ARITH, get EXP
3137  C1         pop bc
3138  D0         ret nc
3139  78         ld a,b
313A  B7         or a
313B  FC FB 35   call m,35FB        REAL ARITH, COMPLEMENT SIGN (ix)
313E  37         scf
313F  C9         ret
```

```
3140  F1          pop af
3141  37          scf
3142  F0          ret p
3143  CD EC 36    call 36EC        ...
3146  AF          xor a
3147  C9          ret

3148  AF          xor a
3149  3C          inc a
314A  C9          ret

314B  4F           ld c,a
314C  F1          pop af
314D  C5          push bc
314E  F5           push af
314F  79            ld a,c
3150  37            scf
3151  8F            adc a,a
3152  30 FD         jr nc,3151     ...
3154  47            ld b,a
3155  CD 0F 33      call 330F      REAL ARITH, copy (hl) to FAC1
3158  EB            ex de,hl
3159  78            ld a,b
315A  87            add a,a
315B  28 15         jr z,3172      ...
315D  F5            push af
315E  CD 1D 33       call 331D     REAL ARITH, copy (hl) to FAC2; mult by (de)
3161  30 16          jr nc,3179    ...
3163  F1            pop af
3164  30 F4         jr nc,315A     ...
3166  F5            push af
3167  11 E8 B8       ld de,B8E8    FAC1
316A  CD 15 34       call 3415     REAL ARITH, MULT, (hl)=(hl)*(de)
316D  30 0A          jr nc,3179    ...
316F  F1            pop af
3170  18 E8         jr 315A        ...

3172  F1           pop af
3173  37           scf
3174  FC FD 32     call m,32FD     get reciprocal value (hl), use FAC3
3177  18 BE        jr 3137         ...

3179  F1            pop af
317A  F1           pop af
317B  C1          pop bc
317C  FA E6 36    jp m,36E6        set exp REAL(ix+4) to 0; set carry; hl=ix
317F  C3 EE 36    jp 36EE          set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF

      @ 3120!
3182  C5          push bc
3183  CD 17 33     call 3317       REAL ARITH, copy (de) to FAC3
3186  CD A1 2E     call 2EA1       REAL ARITH, FIX (hl)
3189  79           ld a,c
318A  C1          pop bc
318B  30 02       jr nc,318F       ...
318D  28 03       jr z,3192        ...
318F  78          ld a,b
3190  B7          or a
3191  C9          ret

3192  4F          ld c,a
3193  7E          ld a,(hl)
3194  1F          rra
3195  9F          sbc a,a
3196  A0          and b
```

```
3197  47            ld b,a
3198  79            ld a,c
3199  FE 02         cp 02           =2.
319B  9F            sbc a,a
319C  D0            ret nc
319D  7E            ld a,(hl)
319E  FE 27         cp 27           =39.
31A0  D8            ret c
31A1  AF            xor a
31A2  C9            ret

----- REAL ARITH, PI (hl)
      @ BD76!
31A3  11 A9 31      ld de,31A9      constant PI = 3.14159265
31A6  C3 18 2E      jp 2E18         copy 5 bytes,(de)>(hl); ld a,(hl-1)

----- constant PI = 3.14159265
31A9  A2 DA 0F 49 82

----- REAL ARITH, set DEG/RAD <a>
      @ BD73!
31AE  32 F7 B8      ld (B8F7),a     flag DEG/RAD
31B1  C9            ret

----- REAL ARITH, COS (hl)
      @ 3235! BD8B!
31B2  CD E8 35      call 35E8       REAL ARITH, SGN (hl); <a>=FF,00,01
31B5  FC FB 35      call m,35FB     REAL ARITH, COMPLEMENT SIGN (ix)
31B8  F6 01         or 01           =1.
31BA  18 01         jr 31BD

----- REAL ARITH, SIN (hl)
      @ 3239! BD88!
31BC  AF            xor a
31BD  F5            push af
31BE  11 1D 32       ld de,321D     constant 0.318309886
31C1  06 F0          ld b,F0        =240.
31C3  3A F7 B8       ld a,(B8F7)    flag DEG/RAD
31C6  B7             or a
31C7  28 05          jr z,31CE      ...
31C9  11 22 32       ld de,3222     constant 5.55556 E-03
31CC  06 F6          ld b,F6        =246.
31CE  CD 07 33       call 3307      test exponent (hl), cp with <b>; set p,z,c
31D1  30 3A          jr nc,320D     ...
31D3  F1            pop af
31D4  CD D5 32      call 32D5       ...
31D7  D0            ret nc
31D8  7B            ld a,e
31D9  1F            rra
31DA  DC FB 35      call c,35FB     REAL ARITH, COMPLEMENT SIGN (ix)
31DD  06 E8         ld b,E8         =232.
31DF  CD 07 33      call 3307       test exponent (hl), cp with <b>; set p,z,c
31E2  D2 E6 36      jp nc,36E6      set exp REAL(ix+4) to 0; set carry; hl=ix
31E5  DD 34 04      inc (ix+04)
31E8  CD A9 32      call 32A9       ...
31EB  06            =6.
31EC  1B 2D 1A E6 6E                constant -3.42879 E-06
31F1  F8 FB 07 28 74                constant 1.60247 E-04
31F6  01 89 68 99 79                constant -4.68165 E-03
31FB  E1 DF 35 23 7D                constant 7.96926 E-02
3200  28 E7 5D A5 80                constant -0645964095
3205  A2 DA 0F 49 81                constant 1.57079633
320A  C3 15 34      jp 3415         REAL ARITH, MULT, (hl)=(hl)*(de)
```

```
320D  F1          pop af
320E  C2 28 33    jp nz,3328        REAL ARITH, copy constant 1. to (hl)
3211  3A F7 B8    ld a,(B8F7)       flag DEG/RAD
3214  FE 01       cp 01             =1.
3216  D8          ret c
3217  11 27 32    ld de,3227        constant 1.74533 E-02
321A  C3 15 34    jp 3415           REAL ARITH, MULT, (hl)=(hl)*(de)

----- constant 0.318309886
321D  6E 83 F9 22 7F

----- constant 5.55556 E-03
3222  B6 60 0B 36 79

----- constant 1.74533 E-02
3227  13 35 FA 0E 7B

----- constant 57.2957795
322C  D3 E0 2E 65 86

----- REAL ARITH, TAN (hl)
      @ BD8E!
3231  CD 0F 33    call 330F         REAL ARITH, copy (hl) to FAC1
3234  D5          push de
3235  CD B2 31     call 31B2        REAL ARITH, COS (hl)
3238  E3           ex (sp),hl
3239  DC BC 31     call c,31BC      REAL ARITH, SIN (hl)
323C  D1          pop de
323D  DA 9E 34    jp c,349E         REAL ARITH DVD; (hl)=(hl)/(de)
3240  C9          ret

----- REAL ARITH, ATN (hl)
      @ BD91!
3241  CD E8 35    call 35E8         REAL ARITH, SGN (hl); <a>=FF,00,01
3244  F5          push af
3245  FC FB 35     call m,35FB      REAL ARITH, COMPLEMENT SIGN (ix)
3248  06 F0        ld b,F0          =240.
324A  CD 07 33     call 3307        test exponent (hl), cp with <b>; set p,z,c
324D  30 4A        jr nc,3299       ...
324F  3D           dec a
3250  F5           push af
3251  F4 FD 32      call p,32FD     get reciprocal value (hl), use FAC3
3254  CD A9 32      call 32A9       ...
3257  0B            =11.
3258  FF C1 03 0F 77                constant 1.09112 E-03
325D  83 FC E8 EB 79                constant 0.007199405
3262  6F CA 78 36 7B                constant 2.22744 E-02
3267  D5 3E B0 B5 7C                constant -4.43575 E-02
326C  B0 C1 8B 09 7D                constant 6.71611 E-02
3271  AF E8 32 B4 7D                constant -8.79877 E-02
3276  74 6C 65 62 7D                constant 0.110545013
327B  D1 F5 37 92 7E                constant -0.142791596
3280  7A C3 CB 4C 7E                constant 0.199996046
3285  83 A7 AA AA 7F                constant 0.333333239
328A  FE FF FF 7F 80                constant 1
328F  CD 15 34 F1 11
3294  05             dec b
3295  32 F4 3B       ld (3BF4),a    ...
3298  33             inc sp
3299  3A F7 B8       ld a,(B8F7)    flag DEG/RAD
329C  B7             or a
329D  11 2C 32       ld de,322C     constant 57.2957795
32A0  C4 15 34       call nz,3415   REAL ARITH, MULT, (hl)=(hl)*(de)
32A3  F1            pop af
32A4  FC FB 35      call m,35FB     REAL ARITH, COMPLEMENT SIGN (ix)
```

```
32A7  37          scf
32A8  C9          ret

      @ 304C! 31E8! 3254!
32A9  CD 1D 33    call 331D        REAL ARITH, copy (hl) to FAC2; mult by (de)

      @ 30BE! 30D2!
32AC  CD 16 33    call 3316        REAL ARITH, copy (hl) to FAC3
32AF  EB          ex de,hl
32B0  D1          pop de           get address of caller
32B1  1A          ld a,(de)        get count of 5 byte sequences
32B2  13          inc de           skip over this parameter
32B3  47          ld b,a
32B4  CD 18 2E    call 2E18        copy 5 bytes,(de)>(hl); ld a,(hl-1)
32B7  13          inc de           skip over 5 byte value
32B8  13          inc de
32B9  13          inc de
32BA  13          inc de
32BB  13          inc de
32BC  D5          push de          restore return address
32BD  11 ED B8     ld de,B8ED      FAC2
32C0  05           dec b
32C1  C8           ret z
32C2  C5           push bc
32C3  11 F2 B8      ld de,B8F2     FAC3
32C6  CD 15 34      call 3415      REAL ARITH, MULT, (hl)=(hl)*(de)
32C9  C1           pop bc
32CA  D1          pop de
32CB  D5          push de
32CC  C5           push bc
32CD  CD 3F 33      call 333F      REAL ARITH, ADD, (hl)=(hl)+(de)
32D0  C1           pop bc
32D1  D1          pop de
32D2  18 E3       jr 32B7          ...

      @ 30AD!
32D4  AF          xor a

      @ 31D4!
32D5  F5          push af
32D6  CD 15 34     call 3415       REAL ARITH, MULT, (hl)=(hl)*(de)
32D9  F1          pop af
32DA  11 CC 30    ld de,30CC       constant 0.5
32DD  C4 3F 33    call nz,333F     REAL ARITH, ADD, (hl)=(hl)+(de)
32E0  E5          push hl
32E1  CD 66 2E     call 2E66       REAL ARITH, CINT; <hl>=int(hl); <a>=sign
32E4  30 13        jr nc,32F9      ...
32E6  D1          pop de
32E7  E5          push hl
32E8  F5           push af
32E9  D5            push de
32EA  11 ED B8       ld de,B8ED    FAC2
32ED  CD 29 2E       call 2E29     REAL ARITH, CREAL <hl> to (de)
32F0  EB             ex de,hl
32F1  E1            pop hl
32F2  CD 37 33      call 3337      REAL ARITH, SUB, (hl)=(hl)-(de)
32F5  F1           pop af
32F6  D1          pop de
32F7  37          scf
32F8  C9          ret
```

```
32F9  E1          pop hl
32FA  AF          xor a
32FB  3C          inc a
32FC  C9          ret

----- get reciprocal value (hl), use FAC3
      @ 3174! 3251!
32FD  CD 16 33    call 3316         REAL ARITH, copy (hl) to FAC3
3300  EB          ex de,hl
3301  CD 28 33    call 3328         REAL ARITH, copy constant 1. to (hl)
3304  C3 9E 34    jp 349E           REAL ARITH DVD; (hl)=(hl)/(de)

----- test exponent (hl), cp with <b>; set p,z,c
      @ 3092! 31CE! 31DF! 324A!
3307  CD 6C 35    call 356C         test expo (hl); z=zero, c=neg; ix=hl
330A  F0          ret p
330B  B8          cp b
330C  C8          ret z
330D  3F          ccf
330E  C9          ret

----- REAL ARITH, copy (hl) to FAC1
      @ 30BA! 3155! 3231!
330F  EB          ex de,hl
3310  21 E8 B8    ld hl,B8E8        FAC1
3313  C3 18 2E    jp 2E18           copy 5 bytes,(de)>(hl); ld a,(hl-1)

----- REAL ARITH, copy (hl) to FAC3
      @ 3035! 32AC! 32FD!
3316  EB          ex de,hl

----- REAL ARITH, copy (de) to FAC3
      @ 3183!
3317  21 F2 B8    ld hl,B8F2        FAC3
331A  C3 18 2E    jp 2E18           copy 5 bytes,(de)>(hl); ld a,(hl-1)

----- REAL ARITH, copy (hl) to FAC2; mult by (de)
      @ 30B7! 315E! 32A9!
331D  EB          ex de,hl
331E  21 ED B8    ld hl,B8ED        FAC2
3321  CD 18 2E    call 2E18         copy 5 bytes,(de)>(hl); ld a,(hl-1)
3324  EB          ex de,hl
3325  C3 15 34    jp 3415           REAL ARITH, MULT, (hl)=(hl)*(de)

----- REAL ARITH, copy constant 1. to (hl)
      @ 3095  3112  320E  3301!
3328  D5          push de
3329  11 32 33     ld de,3332       constant 1.
332C  CD 18 2E     call 2E18        copy 5 bytes,(de)>(hl); ld a,(hl-1)
332F  D1          pop de
3330  37          scf
3331  C9          ret

----- constant 1.
      @ 3039: 3042: 3329:
3332  00 00 00 00 81

----- REAL ARITH, SUB, (hl)=(hl)-(de)
      @ 3045! 30E6! 32F2! BD5B!
3337  3E 01       ld a,01           =1.
3339  18 05       jr 3340
```

```
----- REAL ARITH, SUB (hl)=(de)-(hl)
      @ BD5E!
333B  3E 80      ld a,80        =128.
333D  18 01      jr 3340

----- REAL ARITH, ADD, (hl)=(hl)+(de)
      @ 303C! 307A! 30F1! 32CD! 32DD! BD58!
333F  AF         xor a
3340  E5         push hl
3341  DD E1      pop ix
3343  D5         push de
3344  FD E1      pop iy
3346  DD 46 03   ld b,(ix+03)
3349  FD 4E 03   ld c,(iy+03)
334C  B7         or a
334D  28 0B      jr z,335A      ...
334F  FA 58 33   jp m,3358      ...
3352  3E 80      ld a,80        =128.
3354  A9         xor c
3355  4F         ld c,a
3356  18 02      jr 335A        ...

3358  A8         xor b
3359  47         ld b,a
335A  DD 7E 04   ld a,(ix+04)
335D  FD BE 04   cp (iy+04)
3360  30 14      jr nc,3376     ...
3362  50         ld d,b
3363  41         ld b,c
3364  4A         ld c,d
3365  B7         or a
3366  57         ld d,a
3367  FD 7E 04   ld a,(iy+04)
336A  DD 77 04   ld (ix+04),a
336D  28 54      jr z,33C3      ...
336F  92         sub d
3370  FE 21      cp 21          =33.
3372  30 4F      jr nc,33C3     ...
3374  18 11      jr 3387        ...

3376  AF         xor a
3377  FD 96 04   sub (iy+04)
337A  28 59      jr z,33D5      ...
337C  DD 86 04   add a,(ix+04)
337F  FE 21      cp 21          =33.
3381  30 52      jr nc,33D5     ...
3383  E5         push hl
3384  FD E1      pop iy
3386  EB         ex de,hl
3387  5F         ld e,a
3388  78         ld a,b
3389  A9         xor c
338A  F5         push af
338B  C5          push bc
338C  7B           ld a,e
338D  CD 43 36     call 3643    ...
3390  79           ld a,c
3391  C1          pop bc
3392  4F          ld c,a
3393  F1         pop af
3394  FA DA 33   jp m,33DA      ...
3397  FD 7E 00   ld a,(iy+00)
339A  85         add a,l
339B  6F         ld l,a
339C  FD 7E 01   ld a,(iy+01)
```

```
339F  8C           adc a,h
33A0  67           ld h,a
33A1  FD 7E 02     ld a,(iy+02)
33A4  8B           adc a,e
33A5  5F           ld e,a
33A6  FD 7E 03     ld a,(iy+03)
33A9  CB FF        set 7,a
33AB  8A           adc a,d
33AC  57           ld d,a
33AD  D2 BA 36     jp nc,36BA          ...
33B0  CB 1A        rr d
33B2  CB 1B        rr e
33B4  CB 1C        rr h
33B6  CB 1D        rr l
33B8  CB 19        rr c
33BA  DD 34 04     inc (ix+04)
33BD  C2 BA 36     jp nz,36BA          ...
33C0  C3 EE 36     jp 36EE             set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF

33C3  FD 7E 02     ld a,(iy+02)
33C6  DD 77 02     ld (ix+02),a
33C9  FD 7E 01     ld a,(iy+01)
33CC  DD 77 01     ld (ix+01),a
33CF  FD 7E 00     ld a,(iy+00)
33D2  DD 77 00     ld (ix+00),a
33D5  DD 70 03     ld (ix+03),b
33D8  37           scf
33D9  C9           ret

33DA  AF           xor a
33DB  91           sub c
33DC  4F           ld c,a
33DD  FD 7E 00     ld a,(iy+00)
33E0  9D           sbc a,l
33E1  6F           ld l,a
33E2  FD 7E 01     ld a,(iy+01)
33E5  9C           sbc a,h
33E6  67           ld h,a
33E7  FD 7E 02     ld a,(iy+02)
33EA  9B           sbc a,e
33EB  5F           ld e,a
33EC  FD 7E 03     ld a,(iy+03)
33EF  CB FF        set 7,a
33F1  9A           sbc a,d
33F2  57           ld d,a
33F3  30 16        jr nc,340B          ...
33F5  78           ld a,b
33F6  2F           cpl
33F7  47           ld b,a
33F8  AF           xor a
33F9  91           sub c
33FA  4F           ld c,a
33FB  3E 00        ld a,00             =0.
33FD  9D           sbc a,l
33FE  6F           ld l,a
33FF  3E 00        ld a,00             =0.
3401  9C           sbc a,h
3402  67           ld h,a
3403  3E 00        ld a,00             =0.
3405  9B           sbc a,e
3406  5F           ld e,a
3407  3E 00        ld a,00             =0.
3409  9A           sbc a,d
340A  57           ld d,a
340B  87           add a,a
```

```
340C  DA BA 36    jp c,36BA          ...
340F  C3 B1 36    jp 36B1            ...

----- MULTIPLY by 10., (hl)=(hl)*10.
      @ 2EE8!
3412  11 53 2F    ld de,2F53         constant 10.

----- REAL ARITH, MULT, (hl)=(hl)*(de)
      @ 2F3B  3065! 307E  30E0! 3131! 316A! 320A! 321A  32A0! 32C6! 32D6!
      @ 3325  BD61!
3415  D5          push de
3416  FD E1       pop iy
3418  E5          push hl
3419  DD E1       pop ix
341B  FD 7E 04    ld a,(iy+04)
341E  B7          or a
341F  28 2C       jr z,344D          ...
3421  3D          dec a
3422  CD 48 35    call 3548          ...
3425  28 26       jr z,344D          ...
3427  30 21       jr nc,344A         ...
3429  F5          push af
342A  C5           push bc
342B  CD 50 34      call 3450        ...
342E  79            ld a,c
342F  C1           pop bc
3430  4F           ld c,a
3431  F1          pop af
3432  CB 7A       bit 7,d
3434  20 0D       jr nz,3443         ...
3436  3D          dec a
3437  28 14       jr z,344D          ...
3439  CB 21       sla c
343B  CB 15       rl l
343D  CB 14       rl h
343F  CB 13       rl e
3441  CB 12       rl d
3443  DD 77 04    ld (ix+04),a
3446  B7          or a
3447  C2 BA 36    jp nz,36BA         ...
344A  C3 EE 36    jp 36EE            set REAL(iy) to: FF,FF,FF,(<b> and 7F),FF

344D  C3 E6 36    jp 36E6            set exp REAL(ix+4) to 0; set carry; hl=ix

      @ 342B!
3450  21 00 00    ld hl,0000
3453  5D          ld e,l
3454  54          ld d,h
3455  FD 7E 00    ld a,(iy+00)
3458  CD 93 34    call 3493          ...
345B  FD 7E 01    ld a,(iy+01)
345E  CD 93 34    call 3493          ...
3461  FD 7E 02    ld a,(iy+02)
3464  CD 93 34    call 3493          ...
3467  FD 7E 03    ld a,(iy+03)
346A  F6 80       or 80              =128.
346C  06 08       ld b,08            =8.
346E  1F          rra
346F  4F          ld c,a
3470  30 14       jr nc,3486         ...
3472  7D          ld a,l
3473  DD 86 00    add a,(ix+00)
3476  6F          ld l,a
3477  7C          ld a,h
3478  DD 8E 01    adc a,(ix+01)
```

```
347B  67          ld h,a
347C  7B          ld a,e
347D  DD 8E 02    adc a,(ix+02)
3480  5F          ld e,a
3481  7A          ld a,d
3482  DD 8E 03    adc a,(ix+03)
3485  57          ld d,a
3486  CB 1A       rr d
3488  CB 1B       rr e
348A  CB 1C       rr h
348C  CB 1D       rr l
348E  CB 19       rr c
3490  10 DE       djnz 3470          ...
3492  C9          ret

      @ 3458! 345E! 3464!
3493  B7          or a
3494  20 D6       jr nz,346C         ...
3496  6C          ld l,h
3497  63          ld h,e
3498  5A          ld e,d
3499  57          ld d,a
349A  C9          ret

----- DEVIDE by 10., (hl)=(hl)/10.
      @ 2EF9!
349B  11 53 2F    ld de,2F53         constant 10.

----- REAL ARITH DVD; (hl)=(hl)/(de)
      @ 2F38  3049! 30EB! 323D  3304  BD64!
349E  D5          push de
349F  FD E1       pop iy
34A1  E5          push hl
34A2  DD E1       pop ix
34A4  AF          xor a
34A5  FD 96 04    sub (iy+04)
34A8  28 58       jr z,3502          ...
34AA  CD 48 35    call 3548          ...
34AD  CA E6 36    jp z,36E6          set exp REAL(ix+4) to 0; set carry; hl=ix
34B0  30 4D       jr nc,34FF         ...
34B2  C5          push bc
34B3  4F           ld c,a
34B4  5E           ld e,(hl)
34B5  23           inc hl
34B6  56           ld d,(hl)
34B7  23           inc hl
34B8  7E           ld a,(hl)
34B9  23           inc hl
34BA  66           ld h,(hl)
34BB  6F           ld l,a
34BC  EB           ex de,hl
34BD  FD 46 03     ld b,(iy+03)
34C0  CB F8        set 7,b
34C2  CD 32 35     call 3532         ...
34C5  30 06        jr nc,34CD        ...
34C7  79           ld a,c
34C8  B7           or a
34C9  20 08        jr nz,34D3        ...
34CB  18 31        jr 34FE           ...

34CD  0D           dec c
34CE  29           add hl,hl
34CF  CB 13        rl e
34D1  CB 12        rl d
34D3  DD 71 04     ld (ix+04),c
```

```
34D6  CD 07 35      call 3507         ...
34D9  DD 71 03      ld (ix+03),c
34DC  CD 07 35      call 3507         ...
34DF  DD 71 02      ld (ix+02),c
34E2  CD 07 35      call 3507         ...
34E5  DD 71 01      ld (ix+01),c
34E8  CD 07 35      call 3507         ...
34EB  D4 32 35      call nc,3532      ...
34EE  9F            sbc a,a
34EF  69            ld l,c
34F0  DD 66 01      ld h,(ix+01)
34F3  DD 5E 02      ld e,(ix+02)
34F6  DD 56 03      ld d,(ix+03)
34F9  C1           pop bc
34FA  4F           ld c,a
34FB  C3 BA 36     jp 36BA           ...

34FE  C1           pop bc
34FF  C3 EE 36     jp 36EE           set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF

3502  CD 94 35     call 3594         ...
3505  AF           xor a
3506  C9           ret

      @ 34D6! 34DC! 34E2! 34E8!
3507  0E 01        ld c,01           =1.
3509  38 08        jr c,3513         ...
350B  7A           ld a,d
350C  B8           cp b
350D  3F           ccf
350E  CC 36 35     call z,3536       ...
3511  30 13        jr nc,3526        ...
3513  7D           ld a,l
3514  FD 96 00     sub (iy+00)
3517  6F           ld l,a
3518  7C           ld a,h
3519  FD 9E 01     sbc a,(iy+01)
351C  67           ld h,a
351D  7B           ld a,e
351E  FD 9E 02     sbc a,(iy+02)
3521  5F           ld e,a
3522  7A           ld a,d
3523  98           sbc a,b
3524  57           ld d,a
3525  37           scf
3526  CB 11        rl c
3528  9F           sbc a,a
3529  29           add hl,hl
352A  CB 13        rl e
352C  CB 12        rl d
352E  3C           inc a
352F  20 D8        jr nz,3509        ...
3531  C9           ret

      @ 34C2! 34EB!
3532  7A           ld a,d
3533  B8           cp b
3534  3F           ccf
3535  C0           ret nz

      @ 350E!
3536  7B           ld a,e
3537  FD BE 02     cp (iy+02)
353A  3F           ccf
353B  C0           ret nz
```

```
353C  7C           ld a,h
353D  FD BE 01     cp (iy+01)
3540  3F           ccf
3541  C0           ret nz
3542  7D           ld a,l
3543  FD BE 00     cp (iy+00)
3546  3F           ccf
3547  C9           ret

      @ 3422! 34AA!
3548  4F           ld c,a
3549  DD 7E 03     ld a,(ix+03)
354C  FD AE 03     xor (iy+03)
354F  47           ld b,a
3550  DD 7E 04     ld a,(ix+04)
3553  B7           or a
3554  C8           ret z
3555  81           add a,c
3556  4F           ld c,a
3557  1F           rra
3558  A9           xor c
3559  79           ld a,c
355A  F2 68 35     jp p,3568        ...
355D  DD CB 03 FE  set 7,(ix+03)
3561  D6 7F        sub 7F           =127.
3563  37           scf
3564  C0           ret nz
3565  FE 01        cp 01            =1.
3567  C9           ret

3568  B7           or a
3569  F8           ret m
356A  AF           xor a            =0
356B  C9           ret

----- test expo (hl); z=zero, c=neg; ix=hl
      @ 301F! 3307!
356C  E5           push hl
356D  DD E1        pop ix
356F  DD 7E 04     ld a,(ix+04)
3572  B7           or a
3573  C8           ret z
3574  D6 80        sub 80           =128.
3576  37           scf
3577  C9           ret

----- REAL ARITH, ??
      @ BD67!
3578  E5           push hl
3579  DD E1        pop ix

      @ 30F8
357B  B7           or a
357C  FA 89 35     jp m,3589        ...
357F  DD 86 04     add a,(ix+04)
3582  DD 77 04     ld (ix+04),a
3585  3F           ccf
3586  D8           ret c
3587  18 0B        jr 3594          ...

3589  DD 86 04     add a,(ix+04)
358C  38 02        jr c,3590        ...
358E  AF           xor a
358F  37           scf
3590  DD 77 04     ld (ix+04),a
```

```
3593  C9          ret

      @ 3502! 3587'
3594  DD 46 03    ld b,(ix+03)
3597  CD EE 36    call 36EE        set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF

----- REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01
      @ 302A! 309B! 30A4! BD6A!
359A  E5          push hl
359B  DD E1       pop ix
359D  D5          push de
359E  FD E1       pop iy

----- COMPARE (ix),(iy); <a>=FF,00,01
      @ 2EE1! 2EF4!
35A0  DD 7E 04    ld a,(ix+04)
35A3  FD BE 04    cp (iy+04)
35A6  38 3A       jr c,35E2        ...
35A8  20 33       jr nz,35DD       ...
35AA  B7          or a
35AB  C8          ret z
35AC  DD 7E 03    ld a,(ix+03)
35AF  FD AE 03    xor (iy+03)
35B2  FA DD 35    jp m,35DD        ...
35B5  DD 7E 03    ld a,(ix+03)
35B8  FD 96 03    sub (iy+03)
35BB  20 17       jr nz,35D4       ...
35BD  DD 7E 02    ld a,(ix+02)
35C0  FD 96 02    sub (iy+02)
35C3  20 0F       jr nz,35D4       ...
35C5  DD 7E 01    ld a,(ix+01)
35C8  FD 96 01    sub (iy+01)
35CB  20 07       jr nz,35D4       ...
35CD  DD 7E 00    ld a,(ix+00)
35D0  FD 96 00    sub (iy+00)
35D3  C8          ret z
35D4  9F          sbc a,a
35D5  FD AE 03    xor (iy+03)
35D8  87          add a,a
35D9  9F          sbc a,a
35DA  D8          ret c
35DB  3C          inc a
35DC  C9          ret

35DD  DD 7E 03    ld a,(ix+03)
35E0  18 F6       jr 35D8          ...

35E2  FD 7E 03    ld a,(iy+03)
35E5  2F          cpl
35E6  18 F0       jr 35D8          ...

----- REAL ARITH, SGN (hl); <a>=FF,00,01
      @ 2EB6! 2FA6! 3017! 310E! 3116! 31B2! 3241! BD70!
35E8  E5          push hl
35E9  DD E1       pop ix
35EB  DD 7E 04    ld a,(ix+04)
35EE  B7          or a
35EF  C8          ret z
35F0  DD 7E 03    ld a,(ix+03)
35F3  87          add a,a
35F4  9F          sbc a,a
35F5  D8          ret c
35F6  3C          inc a
35F7  C9          ret
```

```
----- REAL ARITH, COMPLEMENT SIGN (hl)
      @ BD6D!
35F8  E5          push hl
35F9  DD E1       pop ix

----- REAL ARITH, COMPLEMENT SIGN (ix)
      @ 2EBC! 311C! 313B! 31B5! 31DA! 3245! 32A4!
35FB  DD 7E 03    ld a,(ix+03)
35FE  EE 80       xor 80          flip bit 7
3600  DD 77 03    ld (ix+03),a
3603  C9          ret

      @ 2EA6!
3604  AF          xor a
3605  DD 96 04    sub (ix+04)
3608  20 0A       jr nz,3614      ...
360A  06 04       ld b,04         =4.
360C  77          ld (hl),a
360D  23          inc hl
360E  10 FC       djnz 360C       next
3610  0E 01       ld c,01         =1.
3612  37          scf
3613  C9          ret

3614  C6 A0       add a,A0        =160.
3616  D0          ret nc
3617  E5          push hl
3618  CD 3D 36     call 363D      ...
361B  AF           xor a
361C  B8           cp b
361D  8F           adc a,a
361E  B1           or c
361F  4D           ld c,l
3620  44           ld b,h
3621  E1          pop hl
3622  71          ld (hl),c
3623  23          inc hl
3624  70          ld (hl),b
3625  23          inc hl
3626  73          ld (hl),e
3627  23          inc hl
3628  5F          ld e,a
3629  7E          ld a,(hl)
362A  72          ld (hl),d
362B  E6 80       and 80          =128.
362D  47          ld b,a
362E  0E 04       ld c,04         =4.
3630  AF          xor a
3631  B6          or (hl)
3632  20 05       jr nz,3639      ...
3634  2B          dec hl
3635  0D          dec c
3636  20 F9       jr nz,3631      next
3638  0C          inc c
3639  7B          ld a,e
363A  B7          or a
363B  37          scf
363C  C9          ret

      @ 2E76! 3618!
363D  FE 21       cp 21           =33.
363F  38 02       jr c,3643       ...
3641  3E 21       ld a,21         =33.
```

```
      @ 338D! 363F'
3643  5E          ld e,(hl)
3644  23          inc hl
3645  56          ld d,(hl)
3646  23          inc hl
3647  4E          ld c,(hl)
3648  23          inc hl
3649  66          ld h,(hl)
364A  69          ld l,c
364B  EB          ex de,hl
364C  CB FA       set 7,d
364E  01 00 00    ld bc,0000
3651  18 0B       jr 365E         ...

3653  4F          ld c,a
3654  78          ld a,b
3655  B5          or l
3656  47          ld b,a
3657  79          ld a,c
3658  4D          ld c,l
3659  6C          ld l,h
365A  63          ld h,e
365B  5A          ld e,d
365C  16 00       ld d,00         =0.
365E  D6 08       sub 08          =8.
3660  30 F1       jr nc,3653      ...
3662  C6 08       add a,08        =8.
3664  C8          ret z
3665  CB 3A       srl d
3667  CB 1B       rr e
3669  CB 1C       rr h
366B  CB 1D       rr l
366D  CB 19       rr c
366F  3D          dec a
3670  20 F3       jr nz,3665      ...
3672  C9          ret

      @ 36B4!
3673  14          inc d
3674  15          dec d
3675  F8          ret m
3676  20 17       jr nz,368F      ...
3678  57          ld d,a
3679  7B          ld a,e
367A  B4          or h
367B  B5          or l
367C  B1          or c
367D  C8          ret z
367E  7A          ld a,d
367F  D6 08       sub 08          =8.
3681  38 1C       jr c,369F       ...
3683  C8          ret z
3684  53          ld d,e
3685  5C          ld e,h
3686  65          ld h,l
3687  69          ld l,c
3688  0E 00       ld c,00         =0.
368A  14          inc d
368B  15          dec d
368C  28 F1       jr z,367F       ...
368E  F8          ret m
368F  3D          dec a
3690  C8          ret z
3691  CB 21       sla c
3693  CB 15       rl l
```

```
3695  CB 14        rl h
3697  CB 13        rl e
3699  CB 12        rl d
369B  F2 8F 36     jp p,368F         ...
369E  C9           ret

369F  AF           xor a
36A0  C9           ret

----- ix=hl; set exp <b>; get REAL <l><h><e><d>
      @ 2E61!
36A1  E5           push hl
36A2  DD E1        pop ix
36A4  DD 70 04     ld (ix+04),b
36A7  47           ld b,a
36A8  5E           ld e,(hl)
36A9  23           inc hl
36AA  56           ld d,(hl)
36AB  23           inc hl
36AC  7E           ld a,(hl)
36AD  23           inc hl
36AE  66           ld h,(hl)
36AF  6F           ld l,a
36B0  EB           ex de,hl

      @ 2FF7  340F
36B1  DD 7E 04     ld a,(ix+04)
36B4  CD 73 36     call 3673         ...
36B7  DD 77 04     ld (ix+04),a

      @ 33AD  33BD  340C  3447  34FB
36BA  CB 21        sla c
36BC  30 13        jr nc,36D1        ...
36BE  2C           inc l
36BF  20 10        jr nz,36D1        ...
36C1  24           inc h
36C2  20 0D        jr nz,36D1        ...
36C4  1C           inc e
36C5  20 0A        jr nz,36D1        ...
36C7  14           inc d
36C8  20 07        jr nz,36D1        ...
36CA  DD 34 04     inc (ix+04)
36CD  28 1F        jr z,36EE         set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF
36CF  16 80        ld d,80           =128.
36D1  78           ld a,b
36D2  F6 7F        or 7F             =127.
36D4  A2           and d
36D5  DD 77 03     ld (ix+03),a
36D8  DD 73 02     ld (ix+02),e
36DB  DD 74 01     ld (ix+01),h
36DE  DD 75 00     ld (ix+00),l
36E1  DD E5        push ix
36E3  E1           pop hl
36E4  37           scf
36E5  C9           ret

----- set exp REAL(ix+4) to 0; set carry; hl=ix
      @ 30A7  317C  31E2  344D  34AD
36E6  AF           xor a
36E7  DD 77 04     ld (ix+04),a
36EA  18 F5        jr 36E1           ...
```

```
      @ 309E  3143!
36EC  06 00        ld b,00          =0.

----- set REAL(ix) to: FF,FF,FF,(<b> and 7F),FF
      @ 317F  33C0  344A  34FF  3597! 36CD'
36EE  78           ld a,b
36EF  F6 7F        or 7F            =127.
36F1  DD 77 03     ld (ix+03),a
36F4  F6 FF        or FF            =255.
36F6  DD 77 04     ld (ix+04),a
36F9  DD 77 00     ld (ix+00),a
36FC  DD 77 01     ld (ix+01),a
36FF  DD 77 02     ld (ix+02),a
3702  C9           ret

3703  C7 C7 C7 C7 C7

----- INT ARITH, ??
      @ BDA3!
3708  44           ld b,h
3709  CD D1 37     call 37D1        COMPLEMENT <hl> if negative
370C  18 02        jr 3710          ...

----- INT ARITH, BC=0002; E=0
      @ BDA6!
370E  06 00        ld b,00          =0.
3710  1E 00        ld e,00          =0.
3712  0E 02        ld c,02          =2.
3714  C9           ret

----- INT ARITH, unsigned to sign <b>; z=zero, c=+, m=negative
      @ 373F  377D  BA3E! BDA9!
3715  7C           ld a,h
3716  B7           or a
3717  FA 20 37     jp m,3720        ...
371A  B0           or b
371B  FA D4 37     jp m,37D4        INT ARITH, COMPLEMENT <hl>
371E  37           scf
371F  C9           ret

3720  EE 80        xor 80           complement sign
3722  B5           or l
3723  C0           ret nz
3724  78           ld a,b
3725  37           scf
3726  8F           adc a,a
3727  C9           ret

----- INT ARITH ADD; <hl>=<hl>+<de>
      @ BDAC!
3728  B7           or a             clear carry
3729  ED 5A        adc hl,de        hl+de
372B  37           scf
372C  E0           ret po
372D  F6 FF        or FF            =255.
372F  C9           ret

----- INT ARITH, SUB; <hl>=<de>-<hl>
      @ BDB2!
3730  EB           ex de,hl
```

```
----- INT ARITH SUB; <hl>=<hl>-<de>
      @ BDAF!
3731  B7           or a              clear carry
3732  ED 52        sbc hl,de         hl-de
3734  37           scf
3735  E0           ret po
3736  F6 FF        or FF             =255.
3738  C9           ret

----- INT ARITH MUL; <hl>=<hl>*<de>
      @ BDB5!
3739  CD 45 37     call 3745         ...
373C  CD 50 37     call 3750         INT ARITH, ??
373F  D2 15 37     jp nc,3715        INT ARITH, unsigned to sign <b>; z=zero, c=+
3742  F6 FF        or FF             =255.
3744  C9           ret

      @ 3739! 3789!
3745  7C           ld a,h
3746  AA           xor d
3747  47           ld b,a
3748  EB           ex de,hl
3749  CD D1 37     call 37D1         COMPLEMENT <hl> if negative
374C  EB           ex de,hl
374D  C3 D1 37     jp 37D1           COMPLEMENT <hl> if negative

----- INT ARITH, ??
      @ 373C! BDBE!
3750  7C           ld a,h
3751  B7           or a
3752  28 05        jr z,3759         ...
3754  7A           ld a,d
3755  B7           or a
3756  37           scf
3757  C0           ret nz
3758  EB           ex de,hl
3759  B5           or l
375A  C8           ret z
375B  7A           ld a,d
375C  B3           or e
375D  7D           ld a,l
375E  6B           ld l,e
375F  62           ld h,d
3760  C8           ret z
3761  FE 03        cp 03             =3.
3763  38 10        jr c,3775         ...
3765  37           scf
3766  8F           adc a,a
3767  30 FD        jr nc,3766        ...
3769  29           add hl,hl
376A  D8           ret c
376B  87           add a,a
376C  30 02        jr nc,3770        ...
376E  19           add hl,de
376F  D8           ret c
3770  FE 80        cp 80             =128.
3772  20 F5        jr nz,3769        ...
3774  C9           ret

3775  FE 01        cp 01             =1.
3777  C8           ret z
3778  29           add hl,hl
3779  C9           ret
```

```
----- INT ARITH, DVD; <hl>=<hl>/<de>
      @ BDB8!
377A  CD 89 37    call 3789        ...
377D  DA 15 37    jp c,3715        INT ARITH, unsigned to sign <b>; z=zero, c=+
3780  C9          ret

----- INT ARITH, MOD; <hl>=remainder (<hl>/<de>)
      @ BDBB!
3781  4C          ld c,h
3782  CD 89 37    call 3789        ...
3785  EB          ex de,hl
3786  41          ld b,c
3787  18 F4       jr 377D          ...

      @ 377A! 3782!
3789  CD 45 37    call 3745        ...

----- INT ARITH, DVDu <hl>=<hl>/<de>; <de>=remainder
      @ 18C3! BDC1!
378C  7A          ld a,d
378D  B3          or e
378E  C8          ret z
378F  C5          push bc
3790  EB           ex de,hl
3791  06 01        ld b,01         =1.
3793  7C           ld a,h
3794  B7           or a
3795  20 09        jr nz,37A0      ...
3797  7A           ld a,d
3798  BD           cp l
3799  38 05        jr c,37A0       ...
379B  65           ld h,l
379C  2E 00        ld l,00         =0.
379E  06 09        ld b,09         =9.
37A0  7B           ld a,e
37A1  95           sub l
37A2  7A           ld a,d
37A3  9C           sbc a,h
37A4  38 05        jr c,37AB       ...
37A6  04           inc b
37A7  29           add hl,hl
37A8  30 F6        jr nc,37A0      ...
37AA  3F           ccf
37AB  3F           ccf
37AC  78           ld a,b
37AD  44           ld b,h
37AE  4D           ld c,l
37AF  21 00 00     ld hl,0000
37B2  3D           dec a
37B3  20 03        jr nz,37B8      ...
37B5  18 17        jr 37CE         ...

37B7  29          add hl,hl
37B8  F5          push af
37B9  78           ld a,b
37BA  1F           rra
37BB  47           ld b,a
37BC  79           ld a,c
37BD  1F           rra
37BE  4F           ld c,a
37BF  7B           ld a,e
37C0  91           sub c
37C1  7A           ld a,d
37C2  98           sbc a,b
37C3  38 05        jr c,37CA       ...
```

```
37C5  57              ld d,a
37C6  7B              ld a,e
37C7  91              sub c
37C8  5F              ld e,a
37C9  2C              inc l
37CA  F1             pop af
37CB  3D             dec a
37CC  20 E9          jr nz,37B7          ...
37CE  37             scf
37CF  C1             pop bc
37D0  C9             ret

----- COMPLEMENT <hl> if negative
      @ 3709! 3749! 374D
37D1  7C             ld a,h
37D2  B7             or a
37D3  F0             ret p

----- INT ARITH, COMPLEMENT <hl>
      @ 371B  BDC7!
37D4  AF             xor a               =0
37D5  95             sub l               <a>=0-<l>
37D6  6F             ld l,a
37D7  9C             sbc a,h
37D8  95             sub l
37D9  BC             cp h
37DA  67             ld h,a
37DB  37             scf
37DC  C0             ret nz
37DD  FE 01          cp 01               =1.
37DF  C9             ret

----- INT ARITH, get SGN of <hl>; <a>= FF,00,01
      @ BDCA!
37E0  7C             ld a,h
37E1  87             add a,a
37E2  9F             sbc a,a
37E3  D8             ret c
37E4  B5             or l
37E5  C8             ret z
37E6  AF             xor a
37E7  3C             inc a
37E8  C9             ret

----- INT ARITH, COMPARE <hl>,<de>; <a>= FF,00,01
      @ BDC4!
37E9  7C             ld a,h
37EA  AA             xor d
37EB  7C             ld a,h
37EC  F2 F4 37       jp p,37F4           ...
37EF  87             add a,a
37F0  9F             sbc a,a
37F1  D8             ret c
37F2  3C             inc a
37F3  C9             ret

37F4  BA             cp d
37F5  20 F9          jr nz,37F0          ...
37F7  7D             ld a,l
37F8  93             sub e
37F9  20 F5          jr nz,37F0          ...
37FB  C9             ret
```

```
37FC  C7 C7 C7 01


----- SYMBOL images, start of table
      @ 12E3:!
3800  FF C3 C3 C3 C3 C3 C3 FF  FF C0 C0 C0 C0 C0 C0 C0  18 18 18 18 18 18 18 FF
3818  03 03 03 03 03 03 03 FF  0C 18 30 7E 0C 18 30 00  FF C3 E7 DB DB E7 C3 FF
3830  00 01 03 06 CC 78 30 00  3C 66 C3 C3 FF 24 E7 00  00 00 30 60 FF 60 30 00
3848  00 00 0C 06 FF 06 0C 00  18 18 18 18 DB 7E 3C 18  18 3C 7E DB 18 18 18 18
3860  18 5A 3C 99 DB 7E 3C 18  00 03 33 63 FE 60 30 00  3C 66 FF DB DB FF 66 3C
3878  3C 66 C3 DB DB C3 66 3C  FF C3 C3 FF C3 C3 C3 FF  3C 7E DB DB DF C3 66 3C
3890  3C 66 C3 DF DB DB 7E 3C  3C 66 C3 FB DB DB 7E 3C  3C 7E DB DB FB C3 66 3C
38A8  00 01 33 1E CE 7B 31 00  7E 66 66 66 66 66 66 E7  03 03 03 FF 03 03 03 00
38C0  FF 66 3C 18 18 3C 66 FF  18 18 3C 3C 3C 3C 18 18  3C 66 66 30 18 00 18 00
38D8  3C 66 C3 FF C3 C3 66 3C  FF DB DB DB FB C3 C3 FF  FF C3 C3 FB DB DB DB FF
38F0  FF C3 C3 DF DB DB DB FF  FF DB DB DB DF C3 C3 FF  00 00 00 00 00 00 00 00
3908  18 18 18 18 18 00 18 00  6C 6C 6C 00 00 00 00 00  6C 6C FE 6C FE 6C 6C 00
3920  18 3E 58 3C 1A 7C 18 00  00 C6 CC 18 30 66 C6 00  38 6C 38 76 DC CC 76 00
3938  18 18 30 00 00 00 00 00  0C 18 30 30 30 18 0C 00  30 18 0C 0C 0C 18 30 00
3950  00 66 3C FF 3C 66 00 00  00 18 18 7E 18 18 00 00  00 00 00 00 00 18 18 30
3968  00 00 00 7E 00 00 00 00  00 00 00 00 00 18 18 00  06 0C 18 30 60 C0 80 00
3980  7C C6 CE D6 E6 C6 7C 00  18 38 18 18 18 18 7E 00  3C 66 06 3C 60 66 7E 00
3998  3C 66 06 1C 06 66 3C 00  1C 3C 6C CC FE 0C 1E 00  7E 62 60 7C 06 66 3C 00
39B0  3C 66 60 7C 66 66 3C 00  7E 66 06 0C 18 18 18 00  3C 66 66 3C 66 66 3C 00
39C8  3C 66 66 3E 06 66 3C 00  00 00 18 18 00 18 18 00  00 00 18 18 00 18 18 30
39E0  0C 18 30 60 30 18 0C 00  00 00 7E 00 00 7E 00 00  60 30 18 0C 18 30 60 00
39F8  3C 66 66 0C 18 00 18 00  7C C6 DE DE DE C0 7C 00  18 3C 66 66 7E 66 66 00
3A10  FC 66 66 7C 66 66 FC 00  3C 66 C0 C0 C0 66 3C 00  F8 6C 66 66 66 6C F8 00
3A28  FE 62 68 78 68 62 FE 00  FE 62 68 78 68 60 F0 00  3C 66 C0 C0 CE 66 3E 00
3A40  66 66 66 7E 66 66 66 00  7E 18 18 18 18 18 7E 00  1E 0C 0C 0C CC CC 78 00
3A58  E6 66 6C 78 6C 66 E6 00  F0 60 60 60 62 66 FE 00  C6 EE FE FE D6 C6 C6 00
3A70  C6 E6 F6 DE CE C6 C6 00  38 6C C6 C6 C6 6C 38 00  FC 66 66 7C 60 60 F0 00
3A88  38 6C C6 C6 DA CC 76 00  FC 66 66 7C 6C 66 E6 00  3C 66 60 3C 06 66 3C 00
3AA0  7E 5A 18 18 18 18 3C 00  66 66 66 66 66 66 3C 00  66 66 66 66 66 3C 18 00
3AB8  C6 C6 C6 D6 FE EE C6 00  C6 6C 38 38 6C C6 C6 00  66 66 66 3C 18 18 3C 00
3AD0  FE C6 8C 18 32 66 FE 00  3C 30 30 30 30 30 3C 00  C0 60 30 18 0C 06 02 00
3AE8  3C 0C 0C 0C 0C 0C 3C 00  18 3C 7E 18 18 18 18 00  00 00 00 00 00 00 00 FF
3B00  30 18 0C 00 00 00 00 00  00 00 78 0C 7C CC 76 00  E0 60 7C 66 66 66 DC 00
3B18  00 00 3C 66 60 66 3C 00  1C 0C 7C CC CC CC 76 00  00 00 3C 66 7E 60 3C 00
3B30  1C 36 30 78 30 30 78 00  00 00 3E 66 66 3E 06 7C  E0 60 6C 76 66 66 E6 00
3B48  18 00 38 18 18 18 3C 00  06 00 0E 06 06 66 66 3C  E0 60 66 6C 78 6C E6 00
3B60  38 18 18 18 18 18 3C 00  00 00 6C FE D6 D6 C6 00  00 00 DC 66 66 66 66 00
3B78  00 00 3C 66 66 66 3C 00  00 00 DC 66 66 7C 60 F0  00 00 76 CC CC 7C 0C 1E
3B90  00 00 DC 76 60 60 F0 00  00 00 3C 60 3C 06 7C 00  30 30 7C 30 30 36 1C 00
3BA8  00 00 66 66 66 66 3E 00  00 00 66 66 66 3C 18 00  00 00 C6 D6 D6 FE 6C 00
3BC0  00 00 C6 6C 38 6C C6 00  00 00 66 66 66 3E 06 7C  00 00 7E 4C 18 32 7E 00
3BD8  0E 18 18 70 18 18 0E 00  18 18 18 18 18 18 18 00  70 18 18 0E 18 18 70 00
3BF0  76 DC 00 00 00 00 00 00  CC 33 CC 33 CC 33 CC 33  00 00 00 00 00 00 00 00
3C08  F0 F0 F0 F0 00 00 00 00  0F 0F 0F 0F 00 00 00 00  FF FF FF FF 00 00 00 00
3C20  00 00 00 00 F0 F0 F0 F0  F0 F0 F0 F0 F0 F0 F0 F0  0F 0F 0F 0F F0 F0 F0 F0
3C38  FF FF FF FF F0 F0 F0 F0  00 00 00 00 0F 0F 0F 0F  F0 F0 F0 F0 0F 0F 0F 0F
3C50  0F 0F 0F 0F 0F 0F 0F 0F  FF FF FF FF 0F 0F 0F 0F  00 00 00 00 FF FF FF FF
3C68  F0 F0 F0 F0 FF FF FF FF  0F 0F 0F 0F FF FF FF FF  FF FF FF FF FF FF FF FF
3C80  00 00 00 18 18 00 00 00  18 18 18 18 18 00 00 00  00 00 00 1F 1F 00 00 00
3C98  18 18 18 1F 0F 00 00 00  00 00 00 18 18 18 18 18  18 18 18 18 18 18 18 18
3CB0  00 00 00 0F 1F 18 18 18  18 18 18 1F 1F 18 18 18  00 00 00 F8 F8 00 00 00
3CC8  18 18 18 F8 F0 00 00 00  00 00 00 FF FF 00 00 00  18 18 18 FF FF 00 00 00
3CE0  00 00 00 F0 F8 18 18 18  18 18 18 F8 F8 18 18 18  00 00 00 FF FF 18 18 18
3CF8  18 18 18 FF FF 18 18 18  10 38 6C C6 00 00 00 00  0C 18 30 00 00 00 00 00
3D10  66 66 00 00 00 00 00 00  3C 66 60 F8 60 66 FE 00  38 44 BA A2 BA 44 38 00
3D28  7E F4 F4 74 34 34 34 00  1E 30 38 6C 38 18 F0 00  18 18 0C 00 00 00 00 00
3D40  40 C0 44 4C 54 1E 04 00  40 C0 4C 52 44 08 1E 00  E0 10 62 16 EA 0F 02 00
3D58  00 18 18 7E 18 18 7E 00  18 18 00 7E 00 18 18 00  00 00 00 7E 06 06 00 00
3D70  18 00 18 30 66 66 3C 00  18 00 18 18 18 18 18 00  00 00 73 DE CC DE 73 00
3D88  7C C6 C6 FC C6 C6 F8 C0  00 66 66 3C 66 66 3C 00  3C 60 60 3C 66 66 3C 00
```

```
3DA0  00 00 1E 30 7C 30 1E 00  38 6C C6 FE C6 6C 38 00  00 C0 60 30 38 6C C6 00
3DB8  00 00 66 66 66 7C 60 60  00 00 00 FE 6C 6C 6C 00  00 00 00 7E D8 D8 70 00
3DD0  03 06 0C 3C 66 3C 60 C0  03 06 0C 66 66 3C 60 C0  00 E6 3C 18 38 6C C7 00
3DE8  00 00 66 C3 DB DB 7E 00  FE C6 60 30 60 C6 FE 00  00 7C C6 C6 C6 6C EE 00
3E00  18 30 60 C0 80 00 00 00  18 0C 06 03 01 00 00 00  00 00 00 01 03 06 0C 18
3E18  00 00 00 80 C0 60 30 18  18 3C 66 C3 81 00 00 00  18 0C 06 03 03 06 0C 18
3E30  00 00 00 81 C3 66 3C 18  18 30 60 C0 C0 60 30 18  18 30 60 C1 83 06 0C 18
3E48  18 0C 06 83 C1 60 30 18  18 3C 66 C3 C3 66 3C 18  C3 E7 7E 3C 3C 7E E7 C3
3E60  03 07 0E 1C 38 70 E0 C0  C0 E0 70 38 1C 0E 07 03  CC CC 33 33 CC CC 33 33
3E78  AA 55 AA 55 AA 55 AA 55  FF FF 00 00 00 00 00 00  03 03 03 03 03 03 03 03
3E90  00 00 00 00 00 00 FF FF  C0 C0 C0 C0 C0 C0 C0 C0  FF FE FC F8 F0 E0 C0 80
3EA8  FF 7F 3F 1F 0F 07 03 01  01 03 07 0F 1F 3F 7F FF  80 C0 E0 F0 F8 FC FE FF
3EC0  AA 55 AA 55 00 00 00 00  0A 05 0A 05 0A 05 0A 05  00 00 00 00 AA 55 AA 55
3ED8  A0 50 A0 50 A0 50 A0 50  AA 54 A8 50 A0 40 80 00  AA 55 2A 15 0A 05 02 01
3EF0  01 02 05 0A 15 2A 55 AA  00 80 40 A0 50 A8 54 AA  7E FF 99 FF BD C3 FF 7E
3F08  7E FF 99 FF C3 BD FF 7E  38 38 FE FE FE 10 38 00  10 38 7C FE 7C 38 10 00
3F20  6C FE FE FE 7C 38 10 00  10 38 7C FE FE 10 38 00  00 3C 66 C3 C3 66 3C 00
3F38  00 3C 7E FF FF 7E 3C 00  00 7E 66 66 66 66 7E 00  00 7E 7E 7E 7E 7E 7E 00
3F50  0F 07 0D 78 CC CC CC 78  3C 66 66 66 3C 18 7E 18  0C 0C 0C 0C 0C 3C 7C 38
3F68  18 1C 1E 1B 18 78 F8 70  99 5A 24 C3 C3 24 5A 99  10 38 38 38 38 38 7C D6
3F80  18 3C 7E FF 18 18 18 18  18 18 18 18 FF 7E 3C 18  10 30 70 FF FF 70 30 10
3F98  08 0C 0E FF FF 0E 0C 08  00 00 18 3C 7E FF FF 00  00 00 FF FF 7E 3C 18 00
3FB0  80 E0 F8 FE F8 E0 80 00  02 0E 3E FE 3E 0E 02 00  38 38 92 7C 10 28 28 28
3FC8  38 38 10 FE 10 28 44 82  38 38 12 7C 90 28 24 22  38 38 90 7C 12 28 48 88
3FE0  00 3C 18 3C 3C 3C 18 00  3C FF FF 18 0C 18 30 18  18 3C 7E 18 18 7E 3C 18
3FF8  00 24 66 FF 66 24 00 00
```

```
----- himem DEFAULT, SYMBOL AFTER 240.
AB80  18 3C 7E FF 18 18 18 18  18 18 18 18 FF 7E 3C 18  10 30 70 FF FF 70 30 10
AB98  08 0C 0E FF FF 0E 0C 08  00 00 18 3C 7E FF FF 00  00 00 FF FF 7E 3C 18 00
ABB0  80 E0 F8 FE F8 E0 80 00  02 0E 3E FE 3E 0E 02 00  38 38 92 7C 10 28 28 28
ABC8  38 38 10 FE 10 28 44 82  38 38 12 7C 90 28 24 22  38 38 90 7C 12 28 48 88
ABE0  00 3C 18 3C 3C 3C 18 00  3C FF FF 18 0C 18 30 18  18 3C 7E 18 18 7E 3C 18
ABF8  00 24 66 FF 66 24 00 00

----- BASIC flag ??
      @ C012: C025< DF00> F4C4:
AC00  00

----- Indirection: RESET Basic
      @ C064!
AC01  C9 C9 C9

----- Indirection: ERROR MESSAGE
      @ CA94!
AC04  C9 C9 C9

----- Indirection: Undefined token
      @ DDC3!
AC07  C9 C9 C9

----- Indirection: Undefined token after switch
      @ D0A9!
AC0A  C9 C9 C9

----- Indirection: Syntax error
      @ D078!
AC0D  C9 C9 C9

----- Indirection: Line Assembling
      @ DEE1!
AC10  C9 C9 C9

----- Indirection: LIST and EDIT
      @ E196!
AC13  C9 C9 C9

----- Indirection: Get a token while assembling
      @ DF51!
AC16  C9 C9 C9

----- Indirection: Token not found on LIST
      @ E30B!
AC19  C9 C9 C9

----- flag for AUTO
      @ C099> C0DB<
AC1C  00

----- new line number
      @ C0D6< C102>
AC1D  FF FF

----- step for AUTO
      @ C0F7< C121>
AC1F  FF FF

----- output channel number
      @ C1AA: C1BA> C267> C290> C29F> C360> C377>
AC21  00
```

```
----- input channel number
      @ C1B0: C1C0> C424>
AC22  00

----- POS(printer); # of char's written this line
      @ C3C0> C3CB> C3D1< C3DF>
AC23  0F            =15.

----- WIDTH for Printer
      @ C2B3> C339< C3E6<
AC24  C8            =200.

----- POS(tape); # of char's written on this line
      @ C298> C3EC< C3F9:
AC25  FF

----- flag used by FOR
      @ C5EA< C5FD< C673> C69B>
AC26  FF

----- FAC used by FOR
      @ C588: C5E3:
AC27  00 00 00 00 00

----- used by FOR ??
      @ C532< C5D9> C5F1>
AC2C  A6 00

----- used by WHILE, WEND
      @ C74D< C78B< C7AB>
AC2E  FF FF

----- used by ON
      @ C808< C811> C816< C820> C82D> C838> C89E:
AC30  41            [ERR]

      @ C85A:
AC31  FF

      @ C8C4>
AC32  FF FF

----- line# for ON BREAK GOSUB
      @ C84E> C8DA< C909<
AC34  00 00

----- save for Basic PC on BREAK (within event block BREAK)
      @ C847< C8B7>
AC36  11 B5         B511

----- sound chan 1 (bit 0)
      @ C90F: C95E:
AC38  FF FF 00 08 79 C8 FD FF  FF FF FF FF

----- sound chan 2 (bit 1)
      @ C963:
AC44  FF FF 00 08 79 C8 FD FF  FF FF FF FF

----- sound chan 3 (bit 2)
      @ C968:
AC50  FF FF 00 08 79 C8 FD FF  FF FF FF FF
```

```
----- TIMER, block #0  (4 blocks total)
      @ C8F0: C9C0:
AC5C  FF FF FF FF FF FF FF FF  00 02 79 C8 FD FF FF FF  FF FF
AC6E  FF FF FF FF FF FF FF FF  00 04 79 C8 FD FF FF FF  FF FF
AC80  B8 AD FF FF FF FF FF FF  00 08 79 C8 FD FF FF FF  FF FF
AC92  FF FF FF FF FF FF FF FF  00 10 79 C8 FD FF FF FF  FF FF

----- EDIT BUFFER
      In der ersten Haelfte des Edit-Buffers erkennt man die im Direktmodus
      eingegebene Kommandozeile, die den gesamten Speicherinhalt ueber die
      Centronics-Schnittstelle auf den anderen Rechner uebertragen hat.
      @ CA3B: CA43: CA4E: DC5D: DC6E: E164:
ACA4  66 6F 72 20 62 3D 30 20  74 6F 20 36 35 35 33 35      'for b=0 to 65535
ACB4  20 73 74 65 70 20 31 36  3A 3F 23 38 3A 3F 23 38      ' step 16:?#8:?#8
ACC4  2C 68 65 78 24 28 62 2C  34 29 22 20 22 3B 3A 66      ',hex$(b,4)" ";:f
ACD4  6F 72 20 69 3D 20 30 20  74 6F 20 31 35 3A 61 3D      'or i= 0 to 15:a=
ACE4  62 2B 69 3A 67 6F 73 75  62 20 31 36 33 30 3A 3F      'b+i:gosub 1630:?
ACF4  23 38 2C 68 65 78 24 28  78 2C 32 29 22 20 22 3B      '#8,hex$(x,2)" ";
AD04  3A 6E 65 78 74 3A 6E 65  78 74 00 28 78 29 00 24      ':next:next.(x).$
AD14  3D 78 24 3A 47 4F 54 4F  20 31 34 30 30 00 69 2E      '=x$:GOTO 1400.i.
AD24  62 74 73 2B 31 2C 78 3A  69 2E 63 3D 32 3A 69 70      'bts+1,x:i.c=2:ip
AD34  3D 49 4E 53 54 52 28 69  70 2B 31 2C 61 24 2C 62      '=INSTR(ip+1,a$,b
AD44  6C 24 29 3A 49 46 20 69  70 20 54 48 45 4E 20 78      'l$):IF ip THEN x
AD54  3D 56 41 4C 28 22 26 22  2B 4D 49 44 24 28 61 24      '=VAL("&"+MID$(a$
AD64  2C 69 70 2D 32 2C 69 70  2D 31 29 29 3A 50 4F 4B      ',ip-2,ip-1)):POK
AD74  45 20 69 2E 62 74 73 2B  32 2C 78 3A 69 2E 63 3D      'E i.bts+2,x:i.c=
AD84  33 00 70 65 20 65 72 72  6F 72 2C 4E 45 58 54 20      '3.pe error,NEXT
AD94  6D 69 73 73 69 6E 67 00  00 00 00 00 00 00 00 00      'missing.........
ADA4  00 00                                                 '..

----- ERROR ADDRESS (addr where error occurred)
      @ CA8B< CAD3> CADF> CC3B>
ADA6  A7 1D       1DA7

----- program counter on error break
      @ CA9E< CC41>
ADA8  CF 1D       1DCF

----- last Basic ERROR number
      @ C080> C087< CA85< CACD> CC35< D0DD>
ADAA  03          [CINT]          Unexpected RETURN

----- CONTinue pointer
      @ CBBC< CBC1>
ADAB  63 09       0963            this count is <PC on STOP>+1

----- save Basic PC on STOP or END
      @ CBB8< CBCC>
ADAD  62 09       0962            see above

----- ON ERROR address
      @ CAB6> CAFC> CBE0< CBF3<
ADAF  00 00

----- flag ON ERROR
      @ CABA: CB7E> CBDA< CBFB> CC2B> CC38<
ADB1  00

----- SOUND chan-stat
      @ D2C3< D301:
ADB2  00
```

```
----- SOUND vol-env
      @ D2EC<
ADB3  00

----- SOUND ton-env
      @ D2F2<
ADB4  00

----- SOUND period
      @ D2CD<
ADB5  00 00

----- SOUND noise
      @ D2FA<
ADB7  00

----- SOUND volume
      @ D2E4<
ADB8  00

----- SOUND time
      @ D2DA<
ADB9  00 00

----- envelope table address
      @ D35E: D3A0:
ADBB  00

----- SOUND envelope address ??
      @ D3E9:
ADBC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00

----- FAC used by [^] (power)
      @ D4FA:
ADCB  00 00 00 00 00

      @ D5BE:
ADD0  D2 00 03 01 14 00 00 00  00 00 00 00 00 00 00 00  E9 00 00 00 39 00 00 00
ADE8  00 00 DB 00 00 00 56 00  00 00 09 00 4B 00 00 00  00 00 00 00 00 00 FA 00
AE00  00 00 00 00

      @ D5D5<
AE04  2A 00

      @ D5C6:
AE06  4B 00 00 00 D1 1F

----- table of predefined VARTYPES -41 ('A)
      @ D607: (=ADCB+41)
AE0C  05 05 05 05 05 05 05 05  02 02 02 02 02 02 05 05  05 05 05 05 05 05 05 05
AE24  05 05

      @ D88B< D8B2>
AE26  00

----- pointer to BASIC STACK
      @ D713> D77A> D794> D91F> D92C> D934< D964<
AE27  02 AF        AF02

----- pointer to FN subprogram
      @ DA03< DA0C> DA15< DA28> DA31> DA3C< DA5A> DA74>
AE29  00 00        0000
```

```
----- pointer to FN subprogram
      @ D6E0> DA00< DA08> DA2B< DA46<
AE2B  00 00       0000

----- save for semicolon on PRINT
      @ DB8C< DBB3>
AE2D  3B          ';

----- last DATA line#
      @ DD00> DD2B<
AE2E  17 23       2317             8983.

----- pointer to next data
      @ DCEC> DD12<
AE30  59 23       2359             9049.

----- temp storage BASIC STACK pointer
      @ CAA7> DD78<
AE32  FA AE       AEFA

----- program counter on RUN
      @ C835> C88A> C899< CA9B> CB6E> DD71> DD7C<
AE34  CA 1B       1BCA

----- BASIC program counter PC
      @ DD99< DDCE< DDD2> DDD6> DDF1>
AE36  97 1B       1B97

----- flag TRON/TROFF ff/0
      @ DD9D> DDE7<
AE38  00

----- flag used assembling a basic line
      @ DEC3< DF4A< DFC3< DFD3< DFFF> E0A6< E0B8> E0D1<
AE39  00

----- BASIC Program line format
      @ E677< E687> E696< E78F< E882<
AE3A  1D          <next ADDRESS>

----- used by DELETE <line#>, lower addr
      @ E744< E761>
AE3B  FF FF

----- used by DELETE <line#>, upper addr
      @ E74C< E75D>
AE3D  FF FF

----- load pointer while LOAD
      @ EA27< EA30>
AE3F  FF FF

----- LOAD/MERGE flag
      @ EA45< EA84>
AE41  FF

----- LOAD/CHAIN flag
      @ EB95< EB9D> EBA8> EBDA>
AE42  00

----- used by LOAD, CHAIN
      @ EB98< EBC7>
AE43  54 3F       3F54
```

```
----- flag file read protected
      @ C079> C14F< EBDF<
AE45  00

----- number edit buffer (1)
      @ ECF7: ED18: F0DD: F107:
AE46  02 FF FF 03 05 FF FF FF  FF 31 31 31 31                  '.........1111

----- number edit buffer (2)
      @ EDDF: (AE57 @ F127:)
AE53  41 45 34 34 00                                           'AE44.

----- number edit buffer (3)
      @ EDCF:
AE58  90 FF FF FF FF FF 20 39  30 39 31 20 30 30 30 30         '...... 9091 0000

----- number edit buffer (4)
      @ EEAB:
AE68  00 FF FF FF FF FF                                        '......

----- temp store for char
      @ EE9F< EF22< EF39> EF41< EF9B> F036>
AE6E  45           'E

      @ F031> F07F>
AE6F  00

----- number edit buffer index
      @ EEB0< EFD2< F00F> F020< F070> F077<
AE70  63 AE        AE63

----- address of CALLed routine
      @ F1BF< F1E7!
AE72  00 8B        8B00             assembler program to read ROM

----- ROM selection on CALL
      @ F1C4<
AE74  FF           disable upper ROM, disable lower ROM

----- save HL on CALL
      @ F1DB< F1EE>
AE75  D6 1B        1BD6

----- save SP on CALL
      @ F1C7< F1EA>
AE77  FE BF        BFFE             stack pointer for the time being

----- ZONE for TAB
      @ F1F9< F25F>
AE79  0D           =13.

----- flag for PRINT USING
      @ F2DB< F312< F37C> F3A7>
AE7A  FF

----- himem for Basic pointer
      @ C142> D0F5> F4CB< F4FC< F506> F72F> F744> F76B> F7B6<
AE7B  FF 8A        8AFF             last location usable by Basic

----- himem for SYMBOL AFTER pointer
      @ F4D1< F652> F659< F696< F6F4< F71A> F721< F738< F754>
AE7D  7F AB        AB7F             start of user SYMBOL-table -1
```

```
----- low memory boundary pointer
      @ C13E> DEBD> F4D5< F502>
AE7F  40 00        0040

----- start of BASIC program -1 pointer
      @ DCE6> E67A> E7A3> E7C1> E903> E9C1> E9DB> EA9B> EABF> EBBB> EC4A>
      @ F4DD< FB65>
AE81  6F 01        016F

----- end of BASIC program pointer
      @ D5B1> E683< EABB> EAC3< EAD6> EB25> EB2E< EB4E> EB57< EB60> EB80<
      @ EBD7< EC4F> F52C> F530< F571> F7BB> FB75>
AE83  DB 40        40DB             for the time being

----- start of VAR table pointer
      @ D5B4< D5DB> D6C9> D7E8> F533> F537< F549> F574<
AE85  DB 40        40DB

----- start of DIM'd VAR table pointer
      @ D5B7< D5EA> D751> D7F8> D9A0> DAA4> F53A> F53E< F54D> F579<
AE87  E6 41        41E6

----- upper end of DIM'd variables pointer
      @ D5BA< D897> D8DC< D99C> D9E6> D9F5< E70F> F541> F545< F554> F55F>
      @ F58A< F5E6> F5F8> F628> F75F> FC4D>
AE89  E2 61        61E2

----- start of BASIC STACK
      @ F590:
AE8B  00 AA 41 10 00 01 00 01  E4 05 CB 05 6A 06 6A 06  10 00 5E 06 CB 05 06 00
AEA3  B7 08 B0 08 06 B0 41 00  00 FF 7F 90 00 00 00 00  81 01 51 00 00 00 94 00
AEBB  94 00 16 AA 41 0F 00 01  00 01 77 00 00 00 92 00  92 00 10 E1 41 00 00 FF
AED3  7F 90 00 00 00 00 85 01  56 00 00 00 A8 00 A8 00  16 AA 41 0F 00 01 00 01
AEEB  7C 00 00 00 A6 00 A6 00  10 00 91 00 00 00 06 00  00 00 AF 00 00 00 00 C1
AF03  04 00 00 06 2F 90 00 00  0B 2F 88 00 00 00 2F 88  AE 00 00 C1 04 00 00 1A
AF1B  2F 90 00 00 1F 2F 88 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AF33  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AF4B  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AF63  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AF7B  00 00 00 00 00 00 00 00  00 00 00 00 00 00 20 00  20 00 00 00 00 00 00 00
AF93  00 00 00 00 00 00 20 00  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AFAB  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
AFC3  00 00 00 00 00 00 20 00  20 00 00 00 00 00 00 00  00 00 00 00 00 00 20 00
AFDB  20 00 00 02 00 00 00 00  00 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00
AFF3  00 00 00 00 00 00 00 00  20 00 00 00 00

----- not used so far? may be doch!
B000  BF BF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B018  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B030  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B048  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B060  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B078  FF FF FF FF FF FF FF FF  BE 7D FF FF FF FF FF FF  FF FF FF

----- BASIC STACK pointer
      @ C632> C72E> C7B8> DD75> F593< F5A0> F5AC< F5B0> F5B9< FE55>
B08B  FA AE        AEFA

----- low end of used string space pointer
      @ EACB> EBC0> F51F> F55B> F569< F57C> F586< F5CD< F5EA> F5F2<
      @ F622> F62C> F799> F7B2< FB2E> FBEE> FBFA< FC44< FC65> FC9C>
B08D  9A 6F        6F9A
```

```
----- upper bound for string space pointer
      @ F4CE< F523> F5CA> F786> F7AE< FB36> FB6D> FC41>
B08F  FF 8A       8AFF

----- tape buffer flag
      @ F4EB< F63C: F67A: F690<
B091  00

----- pointer to tape buffer, lower end
      @ F64E< F662> F683>
B092  FF 7A       7AFF

----- pointer to tape buffer, upper end
      @ F655< F693>
B094  7F AB       AB7F

----- himem for SYMBOL AFTER (SYS)
      @ F6F1> F71D<
B096  FF AB       ABFF              upmost location for USER SYMBOLS

----- used by GARBAGE COLLECT
      @ F772< F77B> F7C2>
B098  00 10

----- pointer to start of string stack
      @ FBB6< FBBF> FBD6< FC08> FC13< FC7E>
B09A  9C B0       B09C              string stack

----- string stack
      @ B09A: FB7F: FBB3: FC7B:
B09C  01 A1 00 01 9F 6F FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B0B4  FF FF FF FF FF FF

----- temporary string descriptor
      @ F802: FA30< FB4C: FBA2: FBC5: FBD0: FC21:
B0BA  01 A1 00

----- used on GARBAGE COLLECT
      @ FC4A< FC56> FCAB<
B0BD  00 00

----- save on GARBAGE COLLECT
      @ FC50< FCA3> FCAE<
B0BF  7A 54       547A

----- VARTYPE
      @ D6C0> D6D0> D720> D79A> D7C7> D801> D84C> D85E> D885< D8A8> D8FA>
      @ D950< D995< D9D0> DA65> FBBC< FD12> FDF8> FE1A> FE27: FE6C: FE83:
      @ FEA5: FEDA: FF12< FF1D: FF23> FF27> FF2D> FF45> FF4B< FF55> FF67>
B0C1  02

----- Floating point ACU, FAC
      @ CF20> CF6B> D026< E02A> E06E: F245> F249> F2CF> F38B> F398> F846>
      @ F865> F9F1> FABF> FB49> FB59> FBC2< FBDE> FCBC< FD93< FE36: FE40:
      @ FE47> FE4F> FE5C: FE7C< FE96< FEF7< FF0D< FF16: FF34> FF38: FF4E:
      @ FF63:
B0C2  00 8B 00 00 88

----- not used so far
B0C7  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B0DF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
B0F7  FF FF FF FF FF FF FF FF  FF
```

```
----- KL INTERRUPT SERVICE QUEUE
      @ 0066: 00F2< 011D> 0127< 061C:  (B101) @ 00EC>
B100  00 00       0000              empty

----- KL INTERRUPT SERVICE CHAIN
      @ 00F5< 00FE> 0102<
B102  00 B2       B200              SCR FRAME FLY LIST chain

----- KL INTERRUPT SERVICE CLASS
      @ 00E2: 00F8: 0114: 0132: 0142: B950>
B104  00

----- KL save for SP on interrupt service
      @ 010A< 014E>
B105  B8 BF       BFB8              stack pointer for the time being

----- KL private interrupt stack
      @ (B107+80) 00B1: 010E:
B107  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B11F  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B137  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B14F  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B167  00 00 00 00 00 00 00 00  A4 22 31 1F A6 07 EA B1  0F 04 76 0D 12 02 02 B2
B17F  2F 01 C1 F1 00 AF CA AF

----- KL TIME byte 0,1
      @ 009E> 00AC<
B187  01 D3

----- KL TIME byte 2,3
      @ 009A> 00A8<
B189  39 04

----- KL TIME byte 4, (overflows after 116 years)
      @ 00A5<
B18B  00

----- KL FRAME FLY LIST pointer
      @ 00BF> 016A: 0170:
B18C  FE B1       B1FE              SCR FRAME FLY LIST

----- KL FAST TICKER LIST pointer
      @ 00C7> 017D: 0183:
B18E  00 00       0000              empty

----- KL pointer to TICK LIST
      @ 00DC> 0189> 01BF: 01C5:
B190  00 00       0000              empty

----- KL SLOW TICKER COUNT
      @ 00D2:
B192  01          =1.

----- KL SYNC EVENT queue
      @ 0257> 026F< 0288: B928>
B193  00

----- KL SYNC EVENT queue+1
      @ 022B< B921>
B194  00

----- KL EVENT CLASS
      @ 0264> 026C< 0277< 0295: 029B: B932>
B195  00
```

```
----- KL temp store for EXTERNAL COMMAND NAME on search
      @ 0231: 02B2: 030A:
B196  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00

----- KL RSX QUEUE
      @ 02A2> 02A6< 02BF>
B1A6  00 00

----- KL ROM select address
      @ 0080< 034B> B9D6: BA08< BA98: BAA2>
B1A8  00

----- KL contains c006 = start of ROM
      @ 0060> 0086< 0095!
B1A9  06 C0       C006             entry to upper ROM

----- KL ROM state to call
      @ 005D> 0083< BA28:
B1AB  00

----- KL used for rst 3, FAR CALL
      @ B9E7
B1AC  00 00 00 00 00 00 00 00  00 00 00 00 00 00

----- not used so far
B1BA  00 00 00 00 00 00 00 00  00 00 00 00 00 00
```

```
----- SCR screen mode
      @ 0AEC> 0B28<
B1C8  02

----- SCR SCREEN START
      @ 0B40< 0B50> 0B84> 0BDD> 0E24> 0E37>
B1C9  C0

----- SCR offset to screen start
      @ 0B00>
B1CA  04

----- SCR base of RAM for screen
      @ 0AA8< 0B47< 0B53> 0B8D> 0BE6> 0E2C>
B1CB  C0

----- SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl)=scr addr, <b>=ink, <c>=mask
      @ 0C61< 0C68
B1CC  C3 6B 0C    jp 0C6B          SCR PIXELS write, FORCE-mode 0, NEW=INK, (hl

----- SCR current pixel bit map
      @ 0B20: 0BF1: 0C8E> 0CA2> 0F08: 0F18: 0F32: 0F66: 0F7D: 0FA1: 1015:
B1CF  80 40 20 10 08 04 02 01

----- SCR time for flashing period 1
      @ 0CE4< 0CE8> 0D8F>
B1D7  0A          =10.             * 1/50 second

----- SCR time for flashing period 2
      @ 0D88>
B1D8  0A          =10.             * 1/50 second

----- SCR table of colours, flash period 1
      @ 0CD5: 0D8C:
B1D9  0C 04 0A 13 0C 0B 14 15  0D 06 1E 1F 07 12 19 04  17

----- SCR table of colours, flash period 2
      @ 0D32: 0D81:
B1EA  0C 04 0A 13 0C 0B 14 15  0D 06 1E 1F 07 12 19 0A  07

----- SCR flag which flash period is on (1 or 2)
      @ 0CDE< 0D76: 0D84>
B1FB  FF

----- SCR flag
      @ 0D06< 0D7D<
B1FC  00          =0.

----- time count for current flash period
      @ 0D5B: 0D70<
B1FD  07          =7.

----- SCR FRAME FLY LIST
      (0,1)=frame chain; (1,2)=chain; (3)=count;
      (4)= 1B7=Async, 1B0= Near Address; (5,6)=routine addr; (7)=ROM
      @ 0D3C: 0D4F:
B1FE  00 00 00 00 00 81 5B 0D  00

      @ 0FDC< 0FFE>
B207  00 00

----- not used so far
B209  00 00 00
```

```
----- TXT current text stream selected
      @ 10B3< 10B7> 10EA: 1107> 1110<
B20C  00

----- TXT table for text stream parameters (8 times)
      @ 10A5:
B20D  19 00 00 00 00 18 4F FE  02 FF 01 00 91 13 00
B21C  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B22B  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B23A  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B249  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B258  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B267  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00
B276  00 00 00 00 00 18 4F 00  02 FF FF 00 91 13 00

----- TXT CURSOR column/row
      @ 10A8: 1139: 1163> 116E> 117A< 1180> 11AB> 11B1< 133F< 13B1> 1546<
      @ 1560> 1577>
B285  19 00

----- TXT window flag; 0=whole screen
      @ 123E< 125D>
B287  00

----- TXT row; window left upper corner
      @ 116A: 118A> 1197> 11F3> 122D< 1256> 152A> 1543> 1559> 1570>
B288  00

----- TXT column, window left upper corner
      @ 115F: 1190> 119F> 11E1> 11E6> 1533> 1593>
B289  00

----- TXT row, window right bottom corner
      @ 11FB> 1230< 1259> 1549> 155C>
B28A  18

----- TXT column, window right bottom corner
      @ 11DA> 11EE> 1573> 1588>
B28B  4F

----- TXT roll count
      @ 1186> 11B6:
B28C  A2

----- TXT cursor enable flag (user)
      @ 1140< 1263> 1291: 12A2:
B28D  02

----- TXT flag VDU enable
      @ 1335> 1456<
B28E  FF

----- TXT PEN ink
      @ 10CE< 10DE> 126E> 12A9: 12BD> 12C9> 12CF< 1391> 139F> 13C0>
B28F  FF

----- TXT PAPER ink
      @ 10C8< 11C1> 12AE: 12C3> 13D3> 1566> 157D> 1597>
B290  00

----- TXT address of BACK/FOREGROUND routine
      @ 1376> 1383< 1387>
B291  91 13       1391            FOREGROUND
```

```
----- TXT flag graphic char write
      @ 13A7< 140D>
B293  00

----- TXT first char of user matrix table
      @ 1320< 132A>
B294  F0            =240.           default value

----- TXT flag for user matrix table
      @ 107C<
B295  FF

----- TXT start of user matrix table
      @ 1325< 1330>
B296  80 AB         AB80            himem DEFAULT, SYMBOL AFTER 240.

----- TXT buffer for unpacked char matrix
      @ 134E: 13C3: 13E9:
B298  30 30 7C 30 30 36 1C 00  00 00 00 00 00 00 00 00
B2A8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00

----- TXT control code buffer index
      @ 1415: 1447< 145C<
B2B8  00

----- TXT control code buffer for up to 9 parameters
      @ 142E> 143F:
B2B9  0A 00 00 00 00 00 00 00  00 00

----- control code table; <# of parameters>, <routine address>
      @ 1432: 1462: 14CB:
B2C3  00 E2 14    14E2 =0.        just return
B2C6  01 34 13    1334 =1.        TXT WRITE char <a> to screen
B2C9  00 9A 12    129A =0.        TXT CURSOR DISABLE (user)
B2CC  00 89 12    1289 =0.        TXT CURSOR ENABLE (user)
B2CF  01 CA 0A    0ACA =1.        SCR SET MODE <a>
B2D2  01 45 19    1945 =1.        GRA WRITE CHAR <a> at current graphic pos
B2D5  00 51 14    1451 =0.        TXT VDU ENABLE
B2D8  00 D8 14    14D8 =0.        TXT ring the bell
B2DB  00 0A 15    150A =0.        TXT cursor left one step
B2DE  00 0F 15    150F =0.        TXT cursor right one step
B2E1  00 14 15    1514 =0.        TXT cursor down one line
B2E4  00 19 15    1519 =0.        TXT cursor up one line
B2E7  00 40 15    1540 =0.        TXT CLEAR current WINDOW
B2EA  00 30 15    1530 =0.        TXT cursor to start of line
B2ED  01 AE 12    12AE =1.        TXT SET PAPER ink <a>
B2F0  01 A9 12    12A9 =1.        TXT SET PEN ink <a>
B2F3  00 4F 15    154F =0.        TXT clear char at cursor position
B2F6  00 8E 15    158E =0.        TXT clear start of line incl cursor
B2F9  00 84 15    1584 =0.        TXT clear line from cursor to end
B2FC  00 6D 15    156D =0.        TXT clear window from start to cursor
B2FF  00 56 15    1556 =0.        TXT clear window from cursor to end
B302  00 4B 14    144B =0.        TXT VDU DISABLE
B305  01 E3 14    14E3 =1.        TXT set write mode 2, 0=off, 1=on
B308  01 49 0C    0C49 =1.        SCR ACCESS, set write mode <a> for graph VDU
B30B  00 C9 12    12C9 =0.        TXT INVERSE, swap PEN/PAPER ink
B30E  09 04 15    1504 =9.        TXT set matrix for user <symbol>, 8<byte mat
B311  04 F8 14    14F8 =4.        TXT define window, <left>,<right>,<top>,<bot
B314  00 E2 14    14E2 =0.        just return on ESC code
B317  03 E8 14    14E8 =3.        TXT set ink, <PEN>,<colour1>,[<colour2>]
B31A  02 F1 14    14F1 =2.        TXT set border <colour>[<colour>]
B31D  00 2A 15    152A =0.        TXT cursor HOME
B320  02 38 15    1538 =2.        TXT cursor LOCATE <column>(de),<line>(de+1)
```

```
----- not used so far
B323  00 00 00 00 00

----- GRA user origin x
      @ 1604< 1612> 1637>
B328  00 00

----- GRA user origin y
      @ 1608< 1616> 164E>
B32A  00 00

----- GRA cursor x
      @ 15F4< 15FC> 1658>
B32C  00 00

----- GRA cursor y
      @ 15F8< 1600> 165E>
B32E  00 00

----- GRA WINDOW WIDTH, xleft
      @ 1666> 16D0> 16DA> 16DE> 1700> 1758< 17A6> 17E2>
B330  00 00

----- GRA WINDOW WIDTH, xright
      @ 1670> 16C7> 16E8> 16F1> 170A> 175C< 17AA>
B332  7F 02

----- GRA WINDOW HEIGHT, ytop
      @ 167A> 169B> 16A4> 16BD> 1720> 178A< 17BC> 17D9>
B334  C7 00

----- GRA WINDOW HEIGHT, ybottom
      @ 1683> 168D> 1691> 16B3> 1716> 178E< 17C0> 17D5>
B336  00 00

----- GRA PEN INK
      @ 17F9< 1804> 181D> 190A> 192F> 19D3>
B338  FF

----- GRA PAPER ink
      @ 17EC> 1800< 180A> 19D8>
B339  00

----- GRA temp store 1
      @ 1898< 18B3< 1911> 1936> 194A:
B33A  00 00

----- GRA temp store 2
      @ 18C6< 18D8>
B33C  00 00

----- GRA temp store 3
      @ 18C9< 18DC>
B33E  00 00

----- GRA temp store 4
      @ 18BE< 18CD> 18E1>
B340  00 00

----- GRA temp store x on draw
      @ 1841< 184E> 1859> 185D< 18A2> 18A6< 18F7> 18FD< 1927> 193A<
B342  00 00
```

```
----- GRA temp store y on draw
      @ 1845< 1860> 1864< 1872> 18A9> 18AF< 1903> 1915< 191A> 1920<
B344  00 00

----- GRA temp flag
      @ 18BA< 18F1>
B346  00

----- not used so far
B347  00 00 00 00 00

----- KM KEY normal entry
B34C  F0 F3 F1 89 86 83 8B 8A  F2 E0 87 88 85 81 82 80      'psq.....r`......
B35C  10 5B 0D 5D 84 FF 5C FF  5E 2D 40 70 3B 3A 2F 2E      '.[.]..\.^-@p;:/.
B36C  30 39 6F 69 6C 6B 6D 2C  38 37 75 79 68 6A 6E 20      '09oilkm,87uyhjn
B37C  36 35 72 74 67 66 62 76  34 33 65 77 73 64 63 78      '65rtgfbv43ewsdcx
B38C  31 32 FC 71 09 61 FD 7A  0B 0A 08 09 58 5A FF 7F      '12|q.a}z....XZ..

----- KM KEY shift entry
B39C  F4 F7 F5 89 86 83 8B 8A  F6 E0 87 88 85 81 82 80      'twu.....v`......
B3AC  10 7B 0D 7D 84 FF 60 FF  A3 3D 7C 50 2B 2A 3F 3E      '.{.}..`.#=|P+*?>
B3BC  5F 29 4F 49 4C 4B 4D 3C  28 27 55 59 48 4A 4E 20      '_)OILKM<('UYHJN
B3CC  26 25 52 54 47 46 42 56  24 23 45 57 53 44 43 58      '&%RTGFBV$#EWSDCX
B3DC  21 22 FC 51 09 41 FD 5A  0B 0A 08 09 58 5A FF 7F      '!"|Q.A}Z....XZ..

----- KM KEY control entry
B3EC  F8 FB F9 89 86 83 8C 8A  FA E0 87 88 85 81 82 80      'x{y.....z`......
B3FC  10 1B 0D 1D 84 FF 1C FF  1E FF 00 10 FF FF FF FF      '................
B40C  1F FF 0F 09 0C 0B 0D FF  FF FF 15 19 08 0A 0E FF      '................
B41C  FF FF 12 14 07 06 02 16  FF FF 05 17 13 04 03 18      '................
B42C  FF 7E FC 11 E1 01 FE 1A  FF FF FF FF FF FF FF 7F      '.~|.a.~.........

----- KM KEY REPEAT MAP
      @ 19EF:
B43C  07 03 4B FF FF FF FF FF  AB 8F

----- KM function KEY expansion buffer
      @ 1A24:
B446  01 30 01 31 01 32 01 33  01 34 01 35 01 36 01 37      '.0.1.2.3.4.5.6.7
B456  01 38 01 39 01 2E 01 0D  05 52 55 4E 22 0D 00 00      '.8.9.....RUN"...
B466  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B476  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B486  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B496  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B4A6  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B4B6  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B4C6  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00      '................
B4D6  00 00 00 00 00 00 00 00                               '........

----- KM expansion string flag and count
      @ 1A4C> 1A6D<
B4DE  02

----- KM expansion buffer flag
      @ 1AAF< 1ADA:
B4DF  00

----- KM KEYBOARD 'put back' character
      @ 1A43: 1A77<
B4E0  FF          'IGNORE

----- KM pointer to FUNCTION KEY EXPANSION BUFFER
      @ 1A8E< 1B44>
B4E1  46 B4       B446             KM function KEY expansion buffer
```

```
----- KM pointer to end of expansion buffer +1
      @ 1A8A< 1B05>
B4E3  DE B4       B4DE              KM expansion string flag and count

----- KM expansion buffer pointer
      @ 1AAC< 1B00> 1B11> 1B1C< 1B22>  (@ B4E6) 1B27>
B4E5  77 B4       B477

----- KM caps lock state
      @ 19EC< 1B8D: 1BA6> 1BB3>
B4E7  00

----- KM shift lock state
      @ 1B76:
B4E8  00

----- KM KEY repeat speed
      @ 1C15> 1C69> 1C6D<
B4E9  02          =2.               * 1/50 second

----- KM KEY startup delay
      @ 1C4F>
B4EA  1E          =30.              * 1/50 second

----- KM key state map (marks pressed keys by setting the appropriate bit)
      @ 1BCE: 1BFD: 1CC5: (B4ED) @ 1BC6: 1CBE> (B4F1) @ 1C5C> (B4F3)@ 1C2F:
      @ (B4F4) @ 1C62>
B4EB  00 00 00 00 00 00 00 00  00 00

----- KM KEY change state map
      @ 1BBA:
B4F5  FF FF FF FF FF FF FF FF  FF FF

----- KM KEY last cycle state map
      @ 1BB7: 1BCB:  (@ B501) 1BC0>
B4FF  00 00 00 00 00 00 00 00  00 00

----- KM time count for repeat speed
      @ 1BF1: 1C09: 1C18<
B509  05

      @ 1BF6> 1C23<
B50A  00

      @ 19E7<
B50B  40

----- KM BREAK ENABLE FLAG
      @ 1C7E< 1C84: 1C90:
B50C  FF

----- KM event block BREAK
      @ 1C74:
B50D  00 00 00 40 5E C4 FD

      @ 1D39:
B514  00 02 00 02 00 02 00 02  00 02 00 02 00 02 00 02  00 02 00 02 00 02 00 02
B52C  00 02 00 40 00 02 00 02  00 02 00 02 00 02 00 02

      @ 1CEE: 1CFE: 1D26:
B53C  15 0D
```

```
      @ 1D0F: 1D15:
B53E  01 0D

      @ 1C0D> 1D0B: 1D22:
B540  00

----- KM translate normal entry, pointer
      @ 1A01< 1D3E> 1D52>
B541  4C B3       B34C            KM KEY normal entry

----- KM translate shift entry, pointer
      @ 19FD< 1D43> 1D57>
B543  9C B3       B39C            KM KEY shift entry

----- KM translate control entry, pointer
      @ 19F9< 1D48> 1D5C>
B545  EC B3       B3EC            KM KEY control entry

----- KM repeat key, pointer to table
      @ 19F5< 1C02> 1CA6> 1CAE>
B547  3C B4       B43C            KM KEY REPEAT MAP

----- not used so far
B549  00 00 00 00 00 00 00
```

Die 80 Tasten-Nummern sind normal wie folgt belegt:

| | | | | | |
|---|---|---|---|---|---|
| 0 | Cursor up | 27 | p | 54 | b |
| 1 | Cursor right | 28 | ; | 55 | v |
| 2 | Cursor down | 29 | : | 56 | 4 |
| 3 | Function Key '9 | 30 | / | 57 | 3 |
| 4 | Function Key '6 | 31 | . | 58 | e |
| 5 | Function Key '3 | 32 | 0 | 59 | w |
| 6 | Function Key 'ENTER | 33 | 9 | 60 | s |
| 7 | Function Key '. | 34 | o | 61 | d |
| 8 | Cursor left | 35 | i | 62 | c |
| 9 | COPY | 36 | l | 63 | x |
| 10 | Function Key '7 | 37 | k | 64 | 1 |
| 11 | Function Key '8 | 38 | m | 65 | 2 |
| 12 | Function Key '5 | 39 | , | 66 | ESC |
| 13 | Function Key '1 | 40 | 8 | 67 | q |
| 14 | Function Key '2 | 41 | 7 | 68 | TAB |
| 15 | Function Key '0 | 42 | u | 69 | a |
| 16 | CLR | 43 | y | 70 | CAPS LOCK |
| 17 | [ | 44 | h | 71 | z |
| 18 | ENTER | 45 | j | 72 | Joystick 0, up |
| 19 | ] | 46 | n | 73 | Joystick 0, down |
| 20 | Function Key '4 | 47 | SPACE | 74 | Joystick 0, left |
| 21 | SHIFT | 48 | 6 Joystick 1, up | 75 | Joystick 0, right |
| 22 | \ | 49 | 5 Joystick 1, down | 76 | Joystick 0, fire 2 |
| 23 | CTRL | 50 | r Joystick 1, left | 77 | Joystick 0, fire 1 |
| 24 | ^ | 51 | t Joystick 1, right | 78 | not used |
| 25 | - | 52 | g Joystick 1, fire 2 | 79 | DEL |
| 26 | @ | 53 | f Joystick 1, fire 1 | | |

```
----- SOUND flag ??
      @ 1F05: 20B2:
B550  00

----- SOUND save for active sounds
      @ 1E6D< 1EE6> 201F: 20F5:
B551  00

----- SOUND channel bits of active sounds
      @ 1E6A< 1ECB: 1F61: 2283:
B552  00

----- SOUND timer count for 1/100 second
B553  03

----- SOUND rendenzvous byte ??
      @ 1F5B< 1F97:
B554  00

----- SOUND event block
      @ 1E70:
B555  00 00 00 81 03 1F 0C

----- SOUND QUEUE, channel A, (first entry), channels to use
      @ (=B51D+3F) 1E9D: 1EEB: 1F12: 1F48: 1FAD: 1FD2: 2052:
      @ 1E80: 2125:
B55C  00 01 08

----- SOUND queue +3 (tone period)
      @ (=B520+3F) 206F:
B55F  00 00

----- SOUND queue +5 (noise period)
      @ (=B522+3F) 1F74:
B561  EC 00 00 00 00 B2 21 01  B4 21 06 FF 00 00 00 00  00 5A 00 00 01 00 01

      @ (=B539+3F) 208D:
B578  04 00 00 00 00 5A 00 00  0B 14 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B590  00 00 00 00 00 00 00 00  00 00 00

----- SOUND QUEUE, channel B
      @ 212D: 2150:
B59B  01 02 10 00 00 EC 00 00  00 00 B2 21 01 B4 21 06  FF 00 00 00 00 00 5A 00
B5B3  00 01 00 01 04 00 00 00  00 5A 00 00 0B 14 00 00  00 00 00 00 00 00 00 00
B5CB  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00

----- SOUND QUEUE, channel C
      @ 2135: 2148:
B5DA  02 04 20 00 00 EC 00 00  00 00 B2 21 01 B4 21 06  FF 00 00 00 00 00 5A 00
B5F2  00 01 00 01 04 00 00 00  00 5A 00 00 0B 14 00 00  00 00 00 00 00 00 00 00

----- SOUND amplitude envelope
      @ 219A: 2338: 2349:
B60A  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00

      @ 1E7D< 2292:
B619  3F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B631  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B649  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B661  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B679  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B691  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B6A9  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B6C1  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B6D9  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

```
B6F1  00 00 00 00 00 00 00 00  00

----- SOUND tone envelope
      @ 233D: 234E:
B6FA  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B712  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B72A  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B742  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B75A  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B772  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B78A  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B7A2  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B7BA  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B7D2  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00

----- not used so far
B7EA  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00
```

```
----- CAS IN flag; enable prompt message
      @ 238E< 2695> 2760>
B800  00

----- CAS IN flag ??
      @ 269A< 2705: 279F<
B801  00

----- CAS IN file type on read
      @ 2392: 23FC> 2401: 248B: 2528: 256E> 25A9: 27BF>
B802  00

----- CAS IN buffer pointer (lo)
      @ 24CF> 2530< 257D>
B803  00 7B        7B00

----- CAS IN buffer pointer (hi)
      @ 2451> 2456< 24A2> 24A6< 2580<
B805  CF 81        81CF

----- CAS IN filename    HEADER RECORD up to B846
      @ 25D6> 25E1: 25F3:
B807  44 44 49 53 00 00 00 00  00 00 00 00 00 00 00 00        'DDIS............

----- CAS IN block number
      @ 258A:
B817  05           =5.

----- CAS IN last block flag
      @ 253F>
B818  FF

----- CAS IN file type
      @ 23A6>
B819  16           =22.              Ascii file, version 1, not protected

----- CAS IN, data length
      @ 243F> 244A> 244E< 249B> 249F< 24BC> 24D6> 259A<
B81A  00 00        0000

----- CAS IN, data location
      @ 239E> 24B2< 24B9> 24C1< 24D2> 256B>
B81C  00 83        8300

----- CAS IN, first block flag
      @ 2594< 25CA>
B81E  00

----- CAS IN, user fields
      @ 23A2>
B81F  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B837  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

```
----- CAS OUT DIRECT file type on write
      @ 23AB: 2415> 242E: 245F: 24ED: 2667:
B847  00

----- CAS OUT DIRECT, pointer to data (lo)
      @ 2504< 251B> 262C>
B848  00 00

----- CAS OUT DIRECT, pointer to data (hi)
      @ 247F> 2484< 262F<
B84A  00 00

----- CAS OUT DIRECT filename   HEADER RECORD up to B88B
      @ 261E: 2636:
B84C  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00          ................

----- CAS OUT DIRECT block number
      @ 265B:
B85C  00          =0.

----- CAS OUT DIRECT, last block flag
      @ 241F: 264D:
B85D  00

----- CAS OUT DIRECT, file type
      @ 24F9<
B85E  00

----- CAS OUT, len of data, updated while writing
      @ 2469> 2478> 247C< 2507< 2514< 2644> 2658<
B85F  00 00

----- CAS OUT DIRECT, data location
      @ 2632<
B861  00 00

----- CAS OUT DIRECT, first block flag
      @ 2624: 2660<
B863  00

----- CAS OUT DIRECT, total len of data
      @ 24FC<
B864  00 00

----- CAS OUT DIRECT; entry for HEADER
      @ 2500<
B866  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B87E  00 00 00 00 00 00 00 00  00 00 00 00 00 00
```

```
----- CAS OUT filename   HEADER RECORD up to B8CB
      @ 254C: 25DE: 25F6: 2692:
B88C  44 44 49 53 00 00 00 00  00 00 00 00 00 00 00 00        'DDIS............

----- CAS OUT, block number
B89C  04          =4.

----- CAS OUT, file type
      @ 258E>
B89D  FF          =255.
B89E  16          =22.

----- CAS OUT, data length
      @ 2567> 2597>
B89F  CF 06       06CF              ...

----- CAS OUT, data location
B8A1  00 83       8300              ...

----- CAS OUT, first block flag
      @ 25D0>
B8A3  00

----- CAS OUT, user fields (logical length)
B8A4  00 00

----- CAS OUT, user fields (entry addr for machine code program)
      @ 24CA>
B8A6  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
B8BE  00 00 00 00 00 00 00 00  00 00 00 00 00 00

      @ 240C: 2673:
B8CC  00

      @ 2873< 295D: 2973>
B8CD  16

      @ 2956< 29B3>
B8CE  55 42

      @ 2A08> 2A1B<
B8D0  00

----- CAS write speed
      @ 238A<
B8D1  06

      @ 2A0C>
B8D2  53

      @ 28B1< 2990> 29A2< 29A6>
B8D3  0F 1D

----- not used so far
B8D5  00 00 00 00 00 00 00
```

```
----- EDI cursor on flag
      @ 2C1E< 2C35> 2C5B< 2C67> 2DCE<
B8DC  00

----- EDI INSERT/OVERWRITE flag
      @ 2AA5< 2BF9> 2BFD< 2C04>
B8DD  00

----- EDI COPYCURSOR position column/row
      @ 2C72< 2C76> 2C83> 2C94< 2CAC> 2CC4< 2CD5> 2CF0> 2CFE< 2D11> 2D1A<
      @ 2D36>
B8DE  12 0D

----- not used so far
B8E0  00 00 00 00

----- RANDOM NUMBER byte 0,1
      @ 2F9D< 2FAA: 2FC2> 2FD5< 2FE9>
B8E4  07 6C

----- RANDOM NUMBER byte 2,3
      @ 2F97< 2FB8> 2FCD> 2FE2< 2FEC>
B8E6  65 89

----- FAC1
      @ 3167: 3310:
B8E8  00 00 00 00 00

----- FAC2
      @ 32BD: 32EA: 331E:
B8ED  00 00 00 00 00

----- FAC3
      @ 32C3: 3317:
B8F2  00 00 00 00 00

----- flag DEG/RAD
      @ 31AE< 31C3> 3211> 3299>
B8F7  00

-----  not used so far
B8F8  00 00 00 00 00 00 00 00
```

```
----- KL current upper ROM enable, <a>=prevoius ROM state
B900  C3 5E BA     jp BA5E

----- KL current upper ROM disable, <a>=prevoius ROM state
B903  C3 68 BA     jp BA68

----- KL lower ROM enable, <a>=prevoius ROM state
B906  C3 4A BA     jp BA4A

----- KL lower ROM disable, <a>=prevoius ROM state
B909  C3 54 BA     jp BA54

----- KL ROM RESTORE, <a>=previous ROM state
B90C  C3 72 BA     jp BA72

----- KL SELECT an UPPER ROM <c>
B90F  C3 7E BA     jp BA7E

----- KL ask UPPER ROM selection <a>
B912  C3 A2 BA     jp BAA2

----- KL ask CLASS <a> VERSION/MARK <hl> of ROM
B915  C3 83 BA     jp BA83

----- KL restore previous ROM selection, <c>=prev. ROM, <b>=prev. state
B918  C3 8C BA     jp BA8C

----- KL ldir, ROMs disabled
B91B  C3 A6 BA     jp BAA6

----- KL lddr, ROMs disabled
B91E  C3 AC BA     jp BAAC

----- KL POLL SYNCHRONOUS, check for higher priority event
      @ DD7F!
B921  3A 94 B1     ld a,(B194)       KL SYNC EVENT queue+1
B924  B7           or a
B925  C8           ret z
B926  E5           push hl
B927  F3            di
B928  2A 93 B1      ld hl,(B193)     KL SYNC EVENT queue
B92B  7C            ld a,h
B92C  B7            or a

----- KL get ROM address and call
B92D  28 07        jr z,B936         queue empty, return
B92F  23           inc hl
B930  23           inc hl
B931  23           inc hl
B932  3A 95 B1     ld a,(B195)       KL EVENT CLASS
B935  BE           cp (hl)
B936  E1           pop hl
B937  FB           ei
B938  C9           ret

----- rst 7, INTERRUPT ENTRY
      @ 0038
B939  F3           di
B93A  08           ex af,af'
B93B  38 33        jr c,B970         ...
B93D  D9           exx
B93E  79           ld a,c
B93F  37           scf
B940  FB           ei
B941  08           ex af,af'
```

```
B942  F3          di
B943  F5          push af
B944  CB 91        res 2,c
B946  ED 49        out (c),c
B948  CD B1 00     call 00B1       INTERRUPT SERVICE ROUTINE (every 1/300 secon
B94B  B7           or a
B94C  08           ex af,af'
B94D  4F           ld c,a
B94E  06 7F        ld b,7F         VIDEO GATE ARRAY, OUT
B950  3A 04 B1     ld a,(B104)     KL INTERRUPT SERVICE CLASS
B953  B7           or a
B954  28 14        jr z,B96A       it's dormant
B956  FA 6A B9     jp m,B96A       it's disabled
B959  79           ld a,c
B95A  E6 0C        and 0C          =12.
B95C  F5           push af
B95D  CB 91         res 2,c
B95F  D9            exx
B960  CD 0A 01      call 010A      perform asyncronous event(s)
B963  D9            exx
B964  E1           pop hl
B965  79           ld a,c
B966  E6 F3        and F3          =243.
B968  B4           or h
B969  4F           ld c,a
B96A  ED 49        out (c),c
B96C  D9           exx
B96D  F1          pop af
B96E  FB          ei
B96F  C9          ret

B970  08          ex af,af'
B971  E1          pop hl
B972  F5          push af
B973  CB D1        set 2,c
B975  ED 49        out (c),c
B977  CD 3B 00     call 003B       EXTERNAL INTERRUPT
B97A  18 CF        jr B94B         ...

----- = jp(hl), low ROM or RAM, bit 14=lower, 15=upper ROM disabled
      @ 000B
B97C  F3          di
B97D  E5          push hl
B97E  D9           exx
B97F  D1          pop de
B980  18 06       jr B988

----- rst 1 <addr>, LOW JUMP, bit 14=lower, 15=upper ROM disabled
      @ 0008
B982  F3          di
B983  D9          exx
B984  E1          pop hl           get address of caller+1
B985  5E          ld e,(hl)
B986  23          inc hl
B987  56          ld d,(hl)        <de> = <address argument>
B988  08          ex af,af'
B989  7A          ld a,d
B98A  CB BA       res 7,d          remove ROM select bits from address
B98C  CB B2       res 6,d
B98E  07          rlca
B98F  07          rlca
```

```
----- no change of ROM select, 1B6=lower, 1B7=upper ROM disabled
B990  07           rlca
B991  07           rlca
B992  A9           xor c
B993  E6 0C        and 0C
B995  A9           xor c
B996  C5           push bc
B997  CD A8 B9      call B9A8       select ROM and jump to routine
B99A  F3            di
B99B  D9            exx
B99C  08            ex af,af'
B99D  79            ld a,c
B99E  C1           pop bc

----- restore previous ROM state
      @ BA0E'
B99F  E6 03        and 03
B9A1  CB 89        res 1,c
B9A3  CB 81        res 0,c
B9A5  B1           or c
B9A6  18 01        jr B9A9

----- select ROM and jump to routine
      @ B997! B9F9!
B9A8  D5           push de          this is the called <addr>
B9A9  4F            ld c,a
B9AA  ED 49         out (c),c       select the ROM
B9AC  B7            or a
B9AD  08            ex af,af'
B9AE  D9            exx
B9AF  FB            ei
B9B0  C9            ret             'return' to the called <addr>

----- jp(hl), FAR CALL, (hl)=addr, <c>=ROM select
      @ 001B
B9B1  F3           di
B9B2  08           ex af,af'
B9B3  79           ld a,c
B9B4  E5           push hl
B9B5  D9            exx
B9B6  D1           pop de
B9B7  18 15        jr B9CE

----- KL FAR ICALL, jp(hl=param), <addr><ROM state>
      @ 0023  0220
B9B9  F3           di
B9BA  E5           push hl
B9BB  D9            exx
B9BC  E1           pop hl
B9BD  18 09        jr B9C8

----- rst 3, FAR CALL (hl=param), <addr>,<ROM state>
      @ 0018
B9BF  F3           di
B9C0  D9           exx
B9C1  E1           pop hl
B9C2  5E           ld e,(hl)
B9C3  23           inc hl
B9C4  56           ld d,(hl)
B9C5  23           inc hl
B9C6  E5           push hl
B9C7  EB            ex de,hl
B9C8  5E            ld e,(hl)
B9C9  23            inc hl
B9CA  56            ld d,(hl)
```

```
B9CB  23            inc hl
B9CC  08            ex af,af'
B9CD  7E            ld a,(hl)
B9CE  FE FC         cp FC              is another ROM selected?
B9D0  30 BE         jr nc,B990         no change of ROM select, 1B6=lower, 1B7=uppe

      @ BA2C'
B9D2  06 DF        ld b,DF             Expansion ROM select
B9D4  ED 79        out (c),a
B9D6  21 A8 B1     ld hl,B1A8          KL ROM select address
B9D9  46           ld b,(hl)
B9DA  77           ld (hl),a
B9DB  C5           push bc
B9DC  FD E5         push iy
B9DE  3D             dec a
B9DF  FE 07          cp 07
B9E1  30 0F          jr nc,B9F2        ...
B9E3  87             add a,a
B9E4  C6 AC          add a,AC
B9E6  6F             ld l,a
B9E7  CE B1          adc a,B1          <hl>=<hl>+2*<a>+B1AC
B9E9  95             sub l
B9EA  67             ld h,a
B9EB  7E             ld a,(hl)
B9EC  23             inc hl
B9ED  66             ld h,(hl)
B9EE  6F             ld l,a
B9EF  E5             push hl
B9F0  FD E1          pop iy

      @ B9E1'
B9F2  06 7F          ld b,7F           VIDEO GATE ARRAY, OUT
B9F4  79             ld a,c
B9F5  CB D7          set 2,a
B9F7  CB 9F          res 3,a
B9F9  CD A8 B9       call B9A8         select ROM and jump to routine
B9FC  FD E1         pop iy
B9FE  F3            di
B9FF  D9            exx
BA00  08            ex af,af'
BA01  59            ld e,c
BA02  C1           pop bc
BA03  78           ld a,b
BA04  06 DF        ld b,DF             Expansion ROM select
BA06  ED 79        out (c),a
BA08  32 A8 B1     ld (B1A8),a         KL ROM select address
BA0B  06 7F        ld b,7F             VIDEO GATE ARRAY, OUT
BA0D  7B           ld a,e
BA0E  18 8F        jr B99F             restore previous ROM state

----- KL jp(hl) to a sideways ROM
      @ 0013
BA10  F3           di
BA11  E5           push hl
BA12  D9            exx
BA13  D1           pop de
BA14  18 08        jr BA1E

----- rst 2, call to a sideways ROM <addr>, bit 14/15 select the ROM
      @ 0010
BA16  F3           di
BA17  D9           exx
BA18  E1           pop hl
BA19  5E           ld e,(hl)
BA1A  23           inc hl
```

```
BA1B  56           ld d,(hl)
BA1C  23           inc hl
BA1D  E5           push hl
BA1E  08            ex af,af'
BA1F  7A            ld a,d
BA20  CB FA         set 7,d
BA22  CB F2         set 6,d
BA24  E6 C0         and C0
BA26  07            rlca
BA27  07            rlca
BA28  21 AB B1      ld hl,B1AB       KL ROM state to call
BA2B  86            add a,(hl)
BA2C  18 A4         jr B9D2          ...

----- rst 5 <addr>, FIRM JUMP, jump to lower ROM
      @ 0028
BA2E  F3           di
BA2F  D9           exx
BA30  E1           pop hl
BA31  5E           ld e,(hl)
BA32  23           inc hl
BA33  56           ld d,(hl)
BA34  CB 91        res 2,c
BA36  ED 49        out (c),c
BA38  ED 53 3F BA ld (BA3F),de      adjust call address
BA3C  D9           exx
BA3D  FB           ei
BA3E  CD 15 37     call 3715         INT ARITH, unsigned to sign <b>; z=zero, c=+
BA41  F3           di
BA42  D9           exx
BA43  CB D1        set 2,c
BA45  ED 49        out (c),c
BA47  D9           exx
BA48  FB           ei
BA49  C9           ret

----- KL lower ROM enable, <a>=prevoius ROM state
      @ B906
BA4A  F3           di
BA4B  D9           exx
BA4C  79           ld a,c
BA4D  CB 91        res 2,c
BA4F  ED 49        out (c),c
BA51  D9           exx
BA52  FB           ei
BA53  C9           ret

----- KL lower ROM disable, <a>=prevoius ROM state
      @ B909
BA54  F3           di
BA55  D9           exx
BA56  79           ld a,c
BA57  CB D1        set 2,c
BA59  ED 49        out (c),c
BA5B  D9           exx
BA5C  FB           ei
BA5D  C9           ret

----- KL current upper ROM enable, <a>=prevoius ROM state
      @ 05F9! B900  BA7E!
BA5E  F3           di
BA5F  D9           exx
BA60  79           ld a,c
BA61  CB 99        res 3,c
BA63  ED 49        out (c),c
```

```
BA65  D9          exx
BA66  FB          ei
BA67  C9          ret

----- KL current upper ROM disable, <a>=prevoius ROM state
      @ B903
BA68  F3          di
BA69  D9          exx
BA6A  79          ld a,c
BA6B  CB D9       set 3,c
BA6D  ED 49       out (c),c
BA6F  D9          exx
BA70  FB          ei
BA71  C9          ret

----- KL ROM RESTORE, <a>=previous ROM state
      @ B90C  BA8E!
BA72  F3          di
BA73  D9          exx
BA74  A9          xor c
BA75  E6 0C       and 0C
BA77  A9          xor c
BA78  4F          ld c,a
BA79  ED 49       out (c),c
BA7B  D9          exx
BA7C  FB          ei
BA7D  C9          ret

----- KL SELECT an UPPER ROM <c>
      @ 02FF! 0336! B90F  BA83!
BA7E  CD 5E BA    call BA5E         KL current upper ROM enable, <a>=prevoius RO
BA81  18 0F       jr BA92

----- KL ask CLASS <a> VERSION/MARK <hl> of ROM
      @ 02DB! B915
BA83  CD 7E BA    call BA7E         KL SELECT an UPPER ROM <c>
BA86  3A 00 C0    ld a,(C000)       on board ROM
BA89  2A 01 C0    ld hl,(C001)      ROM MARK#, VERSION#, REVISION LEVEL

----- KL restore previous ROM selection, <c>=prev. ROM, <b>=prev. state
      @ 0326  0360  B918
BA8C  F5          push af
BA8D  78           ld a,b
BA8E  CD 72 BA     call BA72        KL ROM RESTORE, <a>=previous ROM state
BA91  F1          pop af
BA92  E5          push hl
BA93  F3           di
BA94  06 DF        ld b,DF          Expansion ROM select
BA96  ED 49        out (c),c
BA98  21 A8 B1     ld hl,B1A8       KL ROM select address
BA9B  46           ld b,(hl)
BA9C  71           ld (hl),c
BA9D  48           ld c,b
BA9E  47           ld b,a
BA9F  FB           ei
BAA0  E1          pop hl
BAA1  C9          ret

----- KL ask UPPER ROM selection <a>
      @ B912
BAA2  3A A8 B1    ld a,(B1A8)       KL ROM select address
BAA5  C9          ret
```

```
----- KL ldir, ROMs disabled
      @ 02B8! 24DE  B91B
BAA6  CD B2 BA    call BAB2        disable ROMs, ldir or lddr, ROMs restore
BAA9  ED B0       ldir
BAAB  C9          ret

----- KL lddr, ROMs disabled
      @ 24E7  B91E
BAAC  CD B2 BA    call BAB2        disable ROMs, ldir or lddr, ROMs restore
BAAF  ED B8       lddr
BAB1  C9          ret

----- disable ROMs, ldir or lddr, ROMs restore
      @ BAA6! BAAC!
BAB2  F3          di
BAB3  D9          exx
BAB4  E1          pop hl           (hl)=either lddr or ldir
BAB5  C5          push bc
BAB6  CB D1        set 2,c
BAB8  CB D9        set 3,c
BABA  ED 49        out (c),c
BABC  CD C7 BA     call BAC7       perform either ldir or lddr
BABF  F3           di
BAC0  D9           exx
BAC1  C1          pop bc
BAC2  ED 49       out (c),c
BAC4  D9          exx
BAC5  FB          ei
BAC6  C9          ret

----- perform either ldir or lddr
      @ BABC!
BAC7  E5          push hl
BAC8  D9           exx
BAC9  FB           ei
BACA  C9           ret

----- rst 4, RAM LAM, ld a,(hl) with ROMs disabled
      @ 0020
BACB  F3          di
BACC  D9          exx
BACD  59          ld e,c
BACE  CB D3       set 2,e
BAD0  CB DB       set 3,e
BAD2  ED 59       out (c),e
BAD4  D9          exx
BAD5  7E          ld a,(hl)
BAD6  D9          exx
BAD7  ED 49       out (c),c
BAD9  D9          exx
BADA  FB          ei
BADB  C9          ret

      @ 28CC! 28F7!
BADC  D9          exx
BADD  79          ld a,c
BADE  F6 0C       or 0C
BAE0  ED 79       out (c),a
BAE2  DD 7E 00    ld a,(ix+00)
BAE5  ED 49       out (c),c
BAE7  D9          exx
BAE8  C9          ret
```

```
BAE9  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF

----- KM INITIALISE key manager
BB00  CF E0 99     rst 1,99E0

----- KM RESET key manager
BB03  CF 1E 9A     rst 1,9A1E

----- KM WAIT CHAR from keyboard =<a>
      @ C433!
BB06  CF 3C 9A     rst 1,9A3C

----- KM READ CHAR from keyboard =<a>
      @ C42C! C439  C43C! C45F!
BB09  CF 42 9A     rst 1,9A42

----- KM RETURN CHAR <a> to 'put back' location
      @ C480!
BB0C  CF 77 9A     rst 1,9A77

----- KM SET EXPANSION string
      @ D44F!
BB0F  CF BD 9A     rst 1,9ABD

----- KM GET EXPANSION string, <a>=exp. token, <l>=char#, =<a>char, =carry
BB12  CF 2E 9B     rst 1,9B2E

----- KM allocate EXP BUFFER (de), <hl>=len
BB15  CF 7B 9A     rst 1,9A7B

----- KM WAIT for KEY
BB18  CF 56 9B     rst 1,9B56

----- KM READ a KEY
BB1B  CF 5C 9B     rst 1,9B5C

----- KM TEST if KEY #<a> is pressed
      @ D415!
BB1E  CF BD 9C     rst 1,9CBD

----- KM GET STATE <h>=caps, <l>=shift lock
BB21  CF B3 9B     rst 1,9BB3

----- KM GET JOYSTICKs 1=<h>, 2=<l>
      @ D423!
BB24  CF 5C 9C     rst 1,9C5C

----- KM SET TRANSLATE entry, <a>=key#, <b>=new translation
      @ D475:
BB27  CF 52 9D     rst 1,9D52

----- KM GET TRANSLATE, in: <a>=key#, out: <a>=translation
BB2A  CF 3E 9D     rst 1,9D3E

----- KM SET SHIFT entry, <a>=key#, <b>=new translation
      @ D47B:
BB2D  CF 57 9D     rst 1,9D57

----- KM GET SHIFT entry, in: <a>=key#, out: <a>=translation
BB30  CF 43 9D     rst 1,9D43

----- KM SET CONTROL entry, <a>=key#, <b>=new translation
      @ D481:
BB33  CF 5C 9D     rst 1,9D5C
```

```
----- KM GET CONTROL entry, in: <a>=key#, out: <a>=translation
BB36  CF 48 9D    rst 1,9D48

----- KM SET REPEAT key# <a>, <b>=0 = not
      @ D470!
BB39  CF AB 9C    rst 1,9CAB

----- KM GET REPEAT key# <a>, nz if repeat
BB3C  CF A6 9C    rst 1,9CA6

----- KM SET DELAY key, <h>=start, <l>=rep. speed
      @ D496:
BB3F  CF 6D 9C    rst 1,9C6D

----- KM GET DELAY key, <h>=start, <l>=rep. speed
BB42  CF 69 9C    rst 1,9C69

----- KM ARM BREAK, (de)=routine, <c>=ROM select
      @ C459!
BB45  CF 71 9C    rst 1,9C71

----- KM DISARM BREAK
      @ C073! C900!
BB48  CF 82 9C    rst 1,9C82

----- KM BREAK EVENT
BB4B  CF 90 9C    rst 1,9C90
```

```
----- TXT INITIALISE text VDU
BB4E  CF 78 90    rst 1,9078

----- TXT RESET text VDU
BB51  CF 88 90    rst 1,9088

----- TXT VDU ENABLE
      @ C38A!
BB54  CF 51 94    rst 1,9451

----- TXT VDU DISABLE
BB57  CF 4B 94    rst 1,944B

----- TXT OUTPUT char or ctl code <a> to VDU
      @ C399
BB5A  CF 00 94    rst 1,9400

----- TXT WRITE char <a> to screen
BB5D  CF 34 93    rst 1,9334

----- TXT READ char from screen <hl>=col/row, =<a>, =carry
BB60  CF AB 93    rst 1,93AB

----- TXT SET GRAPHIC char write, <a>=0=OFF, FF=ON
      @ C324  C387!
BB63  CF A7 93    rst 1,93A7

----- TXT SET WINDOW <hl>=left top, <de>=right bottom corner
      @ C2F8!
BB66  CF 0C 92    rst 1,920C

----- TXT GET WINDOW size, <hl>=left top, <de>=right bottom corner
      @ C2A9!
BB69  CF 56 92    rst 1,9256

----- TXT CLEAR current WINDOW
BB6C  CF 40 95    rst 1,9540

----- TXT SET cursor to COLUMN <a>
BB6F  CF 5E 91    rst 1,915E

----- TXT SET cursor to ROW <a>
BB72  CF 69 91    rst 1,9169

----- TXT SET CURSOR, <hl>=column/row
      @ C2DC!
BB75  CF 74 91    rst 1,9174

----- TXT GET CURSOR position (hl), roll count <a>
      @ C26E! C39E!
BB78  CF 80 91    rst 1,9180

----- TXT CURSOR ENABLE (user)
BB7B  CF 89 92    rst 1,9289

----- TXT CURSOR DISABLE (user)
BB7E  CF 9A 92    rst 1,929A

----- TXT CURSOR ON
      @ C430!
BB81  CF 79 92    rst 1,9279
```

```
----- TXT CURSOR OFF
      @ C436
BB84  CF 81 92     rst 1,9281

----- TXT VALIDATE cursor position <hl> column/row
      @ C271! C3A1!
BB87  CF CE 91     rst 1,91CE

----- TXT PLACE/REMOVE CURSOR on screen
BB8A  CF 68 92     rst 1,9268

----- TXT PLACE/REMOVE CURSOR on screen
BB8D  CF 68 92     rst 1,9268

----- TXT SET PEN ink <a>
      @ C215:
BB90  CF A9 92     rst 1,92A9

----- TXT GET PEN ink, =<a>
BB93  CF BD 92     rst 1,92BD

----- TXT SET PAPER ink <a>
      @ C20D:
BB96  CF AE 92     rst 1,92AE

----- TXT GET PAPER ink =<a>
BB99  CF C3 92     rst 1,92C3

----- TXT INVERSE, swap PEN/PAPER ink
BB9C  CF C9 92     rst 1,92C9

----- TXT SET BACKground being written <a>
BB9F  CF 7A 93     rst 1,937A

----- TXT GET if BACKground is being written <a>
BBA2  CF 87 93     rst 1,9387

----- TXT GET char <a> MATRIX, (hl)=address, carry=user
      @ F6BC!
BBA5  CF D3 92     rst 1,92D3

----- TXT SET char MATRIX, <a>=char, (hl)=matrix to set
BBA8  CF F1 92     rst 1,92F1

----- TXT SET user MATRIX TABLE addr (de), (hl)=new table
      @ F6FD! F726
BBAB  CF FD 92     rst 1,92FD

----- TXT GET user MATRIX TABLE (hl)=addr, <a>=first char in table
      @ F6DD!
BBAE  CF 2A 93     rst 1,932A

----- TXT GET CONTROL code table addr
BBB1  CF CB 94     rst 1,94CB

----- TXT STREAM <a> SELECT, <a>=old text stream
      @ C1A6!
BBB4  CF E8 90     rst 1,90E8

----- TXT SWAP STREAMS <b> with <c>
      @ C30D!
BBB7  CF 07 91     rst 1,9107
```

```
----- GRA INITIALISE graphics VDU
BBBA  CF B0 95    rst 1,95B0

----- GRA RESET
BBBD  CF DF 95    rst 1,95DF

----- GRA MOVE ABSOLUTE, <de>=x, <hl>=y
      @ C505:
BBC0  CF F4 95    rst 1,95F4

----- GRA MOVE RELATIVE, <de>=xd, <hl>=yd
      @ C50A:
BBC3  CF F1 95    rst 1,95F1

----- GRA ASK CURSOR, <de>=x, <hl>=y
      @ D108! D10F!
BBC6  CF FC 95    rst 1,95FC

----- GRA SET ORIGIN, <de>=x, <hl>=y
      @ C4B0!
BBC9  CF 04 96    rst 1,9604

----- GRA GET ORIGIN <de>=x, <hl>=y of user coordinates
BBCC  CF 12 96    rst 1,9612

----- GRA set WINDOW width, <de>=x1, <hl>=x2
      @ C4AA!
BBCF  CF 34 97    rst 1,9734

----- GRA set WINDOW height, <de>=y1, <hl>=y2
      @ C4A4!
BBD2  CF 79 97    rst 1,9779

----- GRA get WINDOW width, <de>=xleft, <hl>=xright>
BBD5  CF A6 97    rst 1,97A6

----- GRA GET WINDOW HEIGHT, <de>=ytop, <hl>=ybottom
BBD8  CF BC 97    rst 1,97BC

----- GRA CLEAR GRAPHIC WINDOW
      @ C4C1!
BBDB  CF C5 97    rst 1,97C5

----- GRA SET PEN, <a>=ink
      @ C4E4!
BBDE  CF F6 97    rst 1,97F6

----- GRA GET PEN, <a>=ink
BBE1  CF 04 98    rst 1,9804

----- GRA SET PAPER, <a>=ink
      @ C4BD!
BBE4  CF FD 97    rst 1,97FD

----- GRA GET PAPER, <a>=ink
BBE7  CF 0A 98    rst 1,980A

----- GRA PLOT ABSOLUTE, <de>=x, <hl>=y
      @ C4D0:
BBEA  CF 13 98    rst 1,9813

----- GRA PLOT RELATIVE, <de>=xd, <hl>=yd
      @ C4D5:
BBED  CF 10 98    rst 1,9810
```

```
----- GRA TEST ABSOLUTE, <de>=x, <hl>=y
      @ C4E9:
BBF0  CF 27 98    rst 1,9827

----- GRA TEST RELATIVE, <de>=xd, <hl>=yd
      @ C4EE:
BBF3  CF 24 98    rst 1,9824

----- GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
      @ C4C6:
BBF6  CF 39 98    rst 1,9839

----- GRA DRAW LINE RELATIVE, <de>=xd, <hl>=yd
      @ C4CB:
BBF9  CF 36 98    rst 1,9836

----- GRA WRITE CHAR <a> at current graphic pos
BBFC  CF 45 99    rst 1,9945
```

```
----- SCR INITIALISE screen pack
BBFF  CF A0 8A    rst 1,8AA0

----- SCR RESET screen pack
BC02  CF B1 8A    rst 1,8AB1

----- SCR SET OFFSET (hl) of screen start
BC05  CF 3C 8B    rst 1,8B3C

----- SCR SET BASE of screen RAM <a>
BC08  CF 45 8B    rst 1,8B45

----- SCR GET LOCATION of screen =<a>offset, =(hl)offset
BC0B  CF 50 8B    rst 1,8B50

----- SCR SET MODE <a>
      @ C255!
BC0E  CF CA 8A    rst 1,8ACA

----- SCR GET MODE <a>, cp 01
BC11  CF EC 8A    rst 1,8AEC

----- SCR CLEAR screen to ink 0
BC14  CF F7 8A    rst 1,8AF7

----- SCR CHAR LIMITS, <b>=columns, <c>=lines
BC17  CF 57 8B    rst 1,8B57

----- SCR CHAR POSITION conv phys coord to screen pos
BC1A  CF 64 8B    rst 1,8B64

----- SCR DOT POSITION convert base coordinates to screen position
BC1D  CF A9 8B    rst 1,8BA9

----- SCR NEXT BYTE, step screen addr (hl) right one byte
BC20  CF F9 8B    rst 1,8BF9

----- SCR PREV BYTE, step screen addr (hl) left one byte
BC23  CF 05 8C    rst 1,8C05

----- SCR NEXT LINE, step screen addr (hl) down one line
BC26  CF 13 8C    rst 1,8C13

----- SCR PREV LINE, step screen addr (hl) up one line
BC29  CF 2D 8C    rst 1,8C2D

----- SCR INK ENCODE, in: <a>=ink#, out: <a>=encoded ink
BC2C  CF 86 8C    rst 1,8C86

----- SCR INK DECODE, in: <a>=encoded ink; out: <a>=ink#
BC2F  CF A0 8C    rst 1,8CA0

----- SCR SET colour of INK, <a>=ink#, <b,c>=colours
      @ C237!
BC32  CF EC 8C    rst 1,8CEC

----- SCR GET colour(s) of INK, =<b,c>
BC35  CF 14 8D    rst 1,8D14

----- SCR SET BORDER, <b,c>=colours
      @ C225!
BC38  CF F1 8C    rst 1,8CF1
```

```
----- SCR GET colour of BORDER
BC3B  CF 19 8D    rst 1,8D19

----- SCR SET FLASHING PERIODS <h,l>
      @ D49D:
BC3E  CF E4 8C    rst 1,8CE4

----- SCR GET FLASHING PERIODS <h,l>
BC41  CF E8 8C    rst 1,8CE8

----- SCR FILL BOX, <a>=ink, <hl,de>=corners
BC44  CF B3 8D    rst 1,8DB3

----- SCR FLOOD BOX, <a>=ink, <hl>=left top, <de>=width/height
BC47  CF B7 8D    rst 1,8DB7

----- SCR CHAR INVERT, (hl)=char pos, <b,c>=inks
BC4A  CF DF 8D    rst 1,8DDF

----- SCR HARDWARE SCROLL, <a>=ink for new line, <b>=0=down, else up
BC4D  CF FA 8D    rst 1,8DFA

----- SCR WINDOW SCROLL up or down
BC50  CF 3E 8E    rst 1,8E3E

----- SCR UNPACK, (hl)=matrix address, (de)=destination
BC53  CF F3 8E    rst 1,8EF3

----- SCR REPACK char matrix to standard
BC56  CF 49 8F    rst 1,8F49

----- SCR ACCESS, set write mode for graph VDU
BC59  CF 49 8C    rst 1,8C49

----- SCR PIXELS write, ignoring write mode
BC5C  CF 6B 8C    rst 1,8C6B

----- SCR HORIZONTAL line plot, <a>=ink, de=xbase, bc=xend, hl=ybase
BC5F  CF C4 8F    rst 1,8FC4

----- SCR VERTICAL line plot, <a>=ink, de=xbase, bc=yend, hl=ybase
BC62  CF 2F 90    rst 1,902F
```

```
----- CAS INITIALISE cassette manager
BC65  CF 70 A3    rst 1,A370

----- CAS SET write SPEED, <hl>=len of half a zero bit, <a>=precompensation
      @ D4D6!
BC68  CF 7F A3    rst 1,A37F

----- CAS NOISY, enable or disable prompt messages <a>
      @ D295
BC6B  CF 8E A3    rst 1,A38E

----- CAS START MOTOR
BC6E  CF 4B AA    rst 1,AA4B

----- CAS STOP MOTOR
BC71  CF 4F AA    rst 1,AA4F

----- CAS RESTORE MOTOR to previous state <a>
BC74  CF 51 AA    rst 1,AA51

----- CAS IN OPEN, (hl)=filename, <b>=len, (de)=2kbuff
      @ D270
BC77  CF 92 A3    rst 1,A392

----- CAS IN CLOSE
      @ D299! EA37!
BC7A  CF FC A3    rst 1,A3FC

----- CAS IN ABANDON
      @ D2B0!
BC7D  CF 01 A4    rst 1,A401

----- CAS IN CHAR from input file
      @ C429  CA56! EB77! EB84! EB88!
BC80  CF 35 A4    rst 1,A435

----- CAS IN DIRECT, read input file into store (hl)
      @ EA33! EBE3!
BC83  CF AB A4    rst 1,A4AB

----- CAS RETURN, put last char read back
      @ C414
BC86  CF 9A A4    rst 1,A49A

----- CAS TEST EOF
      @ C418!
BC89  CF 96 A4    rst 1,A496

----- CAS OUT OPEN, (hl)=filename, <b>=len, (de)=2kbuff
      @ D25C
BC8C  CF AB A3    rst 1,A3AB

----- CAS OUT CLOSE
      @ D2A2!
BC8F  CF 15 A4    rst 1,A415

----- CAS OUT ABANDON
      @ D2B6!
BC92  CF 2E A4    rst 1,A42E

----- CAS OUT CHAR <a> to output file
      @ C40D!
BC95  CF 5B A4    rst 1,A45B
```

```
----- CAS OUT DIRECT, (hl)=data, <de>=len, <a>=type, (bc)=entry addr header
      @ EC7F!
BC98  CF EA A4    rst 1,A4EA

----- CAS CATALOG, (de)= 2k buffer to use
      @ D24E!
BC9B  CF 28 A5    rst 1,A528

----- CAS WRITE a record, (hl)=data, <de>=len, <a>=sync char
BC9E  CF 3F A8    rst 1,A83F

----- CAS READ a record, (hl)=data, <de>=len, <a>=expected sync
BCA1  CF 36 A8    rst 1,A836

----- CAS CHECK tape with store, (hl)=data, <de>=len, <a>=sync char
BCA4  CF 51 A8    rst 1,A851
```

```
----- SOUND RESET
      @ C8ED!
BCA7  CF 68 9E    rst 1,9E68

----- SOUND QUEUE, add a sound, (hl)=sound program
      @ D304!
BCAA  CF 9F 9F    rst 1,9F9F

----- SOUND CHECK for space in <a>, <a>=status
      @ D33B!
BCAD  CF 6C A0    rst 1,A06C

----- SOUND ARM EVENT, <a>=channels, (hl)=event block
      @ C958!
BCB0  CF 89 A0    rst 1,A089

----- SOUND RELEASE, <a>=channel(s)
      @ D324!
BCB3  CF 4A A0    rst 1,A04A

----- SOUND HOLD, stop all sounds
      @ C070! C46F!
BCB6  CF CB 9E    rst 1,9ECB

----- SOUND CONTINUE stopped sounds
      @ C484! CBD2!
BCB9  CF E6 9E    rst 1,9EE6

----- SOUND set AMPL ENVELOPE, <a>=env#, (hl)=data
      @ D362!
BCBC  CF 38 A3    rst 1,A338

----- SOUND set TONE ENVELOPE, <a>=env#, (hl)=data
      @ D3A9!
BCBF  CF 3D A3    rst 1,A33D

----- SOUND get AMPL ENV ADDR, <a>=env#, (hl)=addr
BCC2  CF 49 A3    rst 1,A349

----- SOUND get TONE ENV ADDR, <a>=env#, (hl)=addr
BCC5  CF 4E A3    rst 1,A34E
```

```
----- KL CHOKE OFF, reset the kernel
BCC8  CF 5C 80    rst 1,805C

----- KL ROM WALK, (de)=low, (hl)=hi avail. memory
      @ C009!
BCCB  CF 29 83    rst 1,8329

----- KL INIT BACKground ROM, <c>=ROM sel, <de>=lomem, <hl>=himem
BCCE  CF 32 83    rst 1,8332

----- KL LOG EXT, (bc)=RSX cmd table, (hl)=4 byte RAM area
BCD1  CF A1 82    rst 1,82A1

----- KL FIND COMMAND (hl) in RSX or back ROM, =<c>ROM sel, =(hl)routine
      @ F1A7!
BCD4  CF B2 82    rst 1,82B2

----- KL NEW FRAME FLY, (hl)=addr, <b>=class, <de,c>=far addr
BCD7  CF 63 81    rst 1,8163

----- KL ADD FRAME FLY; (hl)=addr of block
BCDA  CF 6A 81    rst 1,816A

----- KL DEL FRAME FLY, remove a block (hl) from the list
BCDD  CF 70 81    rst 1,8170

----- KL NEW FAST TICKER, (hl)=block,<b>=class,<c>=ROM sel,(de)=event routine
BCE0  CF 76 81    rst 1,8176

----- KL ADD FAST TICKER, put block (hl) onto list
BCE3  CF 7D 81    rst 1,817D

----- KL DEL FAST TICKER, remove block (hl) from the list
BCE6  CF 83 81    rst 1,8183

----- KL ADD TICKER, (hl)=tick block, <de>=initial count, <bc>=recharge value
      @ C99A!
BCE9  CF B3 81    rst 1,81B3

----- KL DEL TICKER, remove block (hl) from tick list
      @ C8F6! C9A5!
BCEC  CF C5 81    rst 1,81C5

----- KL INIT EVENT BLOCK (hl)=block, <b>=class, <c>=ROM sel, (de)=routine
      @ C92B!
BCEF  CF D2 81    rst 1,81D2

----- KL EVENT, kick an event block (hl)
BCF2  CF E2 81    rst 1,81E2

----- KL SYNC RESET, clear synchronous event queue
      @ C903!
BCF5  CF 28 82    rst 1,8228

----- KL DEL SYNC, delete block (hl) from queue
BCF8  CF 85 82    rst 1,8285

----- KL NEXT SYNC, =(hl), =<a> prev. prio, =carry
      @ C80B!
BCFB  CF 56 82    rst 1,8256

----- KL DO SYNC, perform SYNC EVENT block (hl)
      @ C81B!
BCFE  CF 1A 82    rst 1,821A
```

```
----- KL DONE SYNC, (hl)=block, <a>=prev. priority
      @ C826! C8AE! C8BE!
BD01  CF 77 82    rst 1,8277

----- KL EVENT DISABLE
      @ C8E2!
BD04  CF 95 82    rst 1,8295

----- KL EVENT ENABLE
      @ C8E8!
BD07  CF 9B 82    rst 1,829B

----- KL DISARM EVENT block (hl)
BD0A  CF 8E 82    rst 1,828E

----- KL TIME PLEASE in <de,hl>
      @ D0E6!
BD0D  CF 99 80    rst 1,8099

----- KL TIME SET <de,hl>
BD10  CF A3 80    rst 1,80A3
```

```
----- MC BOOT PROGRAM, load and run FOREGROUND
      @ E9D5
BD13  CF DC 85    rst 1,85DC

----- MC START FOREGROUND PROGRAM, (hl)=entry addr, <c>=ROM selection
BD16  CF 0B 86    rst 1,860B

----- MC WAIT FLYBACK
BD19  CF BA 87    rst 1,87BA

----- MC SET SCREEN MODE <a>
BD1C  CF 76 87    rst 1,8776

----- MC set SCREEN OFFSET, <a>=base, <hl>=offset
BD1F  CF C6 87    rst 1,87C6

----- MC CLEAR INKS to one colour, (de)=ink vector
BD22  CF 86 87    rst 1,8786

----- MC SET INKS, (de)=ink vector
BD25  CF 99 87    rst 1,8799

----- MC RESET PRINTER indirection
BD28  CF E6 87    rst 1,87E6

----- MC PRINT CHAR <a> to Centronics port
      @ C3D6!
BD2B  CF F2 87    rst 1,87F2

----- MC BUSY PRINTER, if port is busy, =carry
BD2E  CF 1B 88    rst 1,881B

----- MC SEND char <a> to PRINTER
BD31  CF 07 88    rst 1,8807

----- MC SOUND REGISTER, send <a>=reg#, <c>=data
BD34  CF 26 88    rst 1,8826

----- JUMP RESTORE standard jumpblock
BD37  CF 88 88    rst 1,8888

----- EDI LINE EDITOR (hl)
      @ CA40  CA46!
BD3A  CF 98 AA    rst 1,AA98  2A98 EDI LINE EDITOR (hl)

----- copy 5 bytes,(de)>(hl); ld a,(hl-1)
      @ D4FD! ED3A!
BD3D  EF 18 2E    rst 5,2E18

----- REAL ARITH, CREAL <hl> to (de)
      @ FE79
BD40  EF 29 2E    rst 5,2E29

----- REAL ARITH, CREAL (hl) 4 byte integer to real
      @ FDDC! FE03  FE8A
BD43  EF 55 2E    rst 5,2E55

----- REAL ARITH, CINT; <hl>=int(hl); <a>=sign
      @ FEB4! FEC6!
BD46  EF 66 2E    rst 5,2E66

----- REAL ARITH ??
      @ FDC9: FDD4!
BD49  EF 8E 2E    rst 5,2E8E
```

```
----- REAL ARITH, FIX (hl)
      @ FDE8:
BD4C  EF A1 2E    rst 5,2EA1

----- REAL ARITH, INT (hl)
      @ FDED:
BD4F  EF AC 2E    rst 5,2EAC

----- REAL ARITH ??
      @ FCB6
BD52  EF B6 2E    rst 5,2EB6

----- REAL ARITH ??
      @ ED33! FDD1! FDE1
BD55  EF 1D 2F    rst 5,2F1D

----- REAL ARITH, ADD, (hl)=(hl)+(de)
      @ FCDA!
BD58  EF 3F 33    rst 5,333F

----- REAL ARITH, SUB, (hl)=(hl)-(de)
BD5B  EF 37 33    rst 5,3337

----- REAL ARITH, SUB (hl)=(de)-(hl)
      @ FCEF!
BD5E  EF 3B 33    rst 5,333B

----- REAL ARITH, MULT, (hl)=(hl)*(de)
      @ FD03!
BD61  EF 15 34    rst 5,3415

----- REAL ARITH DVD; (hl)=(hl)/(de)
      @ FD24!
BD64  EF 9E 34    rst 5,349E

----- REAL ARITH, ??
BD67  EF 78 35    rst 5,3578

----- REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01
      @ FD0F
BD6A  EF 9A 35    rst 5,359A

----- REAL ARITH, COMPLEMENT SIGN (hl)
      @ FD9D!
BD6D  EF F8 35    rst 5,35F8

----- REAL ARITH, SGN (hl); <a>=FF,00,01
      @ D597! FDAA!
BD70  EF E8 35    rst 5,35E8

----- REAL ARITH, set DEG/RAD <a>
      @ C15F! D4EC
BD73  EF AE 31    rst 5,31AE

----- REAL ARITH, PI (hl)
      @ D4E2!
BD76  EF A3 31    rst 5,31A3

----- REAL ARITH, SQR (hl) ^ 0.5
      @ D4EF:
BD79  EF 0A 31    rst 5,310A
```

```
----- REAL ARITH, EXP; (hl)=(hl)^(de)
      @ D507:
BD7C  EF 0D 31    rst 5,310D

----- REAL ARITH, LOG (hl)
      @ D52A:
BD7F  EF 14 30    rst 5,3014

----- REAL ARITH, LOG10 (hl)
      @ D525:
BD82  EF 0F 30    rst 5,300F

----- REAL ARITH, get EXP
      @ D520:
BD85  EF 90 30    rst 5,3090

----- REAL ARITH, SIN (hl)
      @ D52F:
BD88  EF BC 31    rst 5,31BC

----- REAL ARITH, COS (hl)
      @ D534:
BD8B  EF B2 31    rst 5,31B2

----- REAL ARITH, TAN (hl)
      @ D539:
BD8E  EF 31 32    rst 5,3231

----- REAL ARITH, ATN (hl)
      @ D53E:
BD91  EF 41 32    rst 5,3241

----- REAL ARITH ??
      @ ED2F!
BD94  EF 5E 2E    rst 5,2E5E

----- REAL ARITH, set initial RANDOM NUMBER
      @ C02E!
BD97  EF 94 2F    rst 5,2F94

----- REAL ARITH, seed RANDOM NUMBER
      @ D57F! D5A1!
BD9A  EF A1 2F    rst 5,2FA1

----- REAL ARITH, RANDOMIZE
      @ D5A9!
BD9D  EF B7 2F    rst 5,2FB7

----- REAL ARITH, get last RANDOM NUMBER (hl)
      @ D59C!
BDA0  EF E6 2F    rst 5,2FE6

----- INT ARITH, ??
      @ FCB9!
BDA3  EF 08 37    rst 5,3708

----- INT ARITH, BC=0002; E=0
      @ FCC9
BDA6  EF 0E 37    rst 5,370E

----- INT ARITH, unsigned to sign <b>; z=zero, c=+, m=negative
      @ FE0E! FEB8! FECC!
BDA9  EF 15 37    rst 5,3715
```

```
----- INT ARITH ADD; <hl>=<hl>+<de>
      @ C6A9! FCD1!
BDAC  EF 28 37    rst 5,3728

----- INT ARITH SUB; <hl>=<hl>-<de>
BDAF  EF 31 37    rst 5,3731

----- INT ARITH, SUB; <hl>=<de>-<hl>
      @ FCE6!
BDB2  EF 30 37    rst 5,3730

----- INT ARITH MUL; <hl>=<hl>*<de>
      @ FCFA!
BDB5  EF 39 37    rst 5,3739

----- INT ARITH, DVD; <hl>=<hl>/<de>
      @ FD3B!
BDB8  EF 7A 37    rst 5,377A

----- INT ARITH, MOD; <hl>=remainder (<hl>/<de>)
      @ FD4D!
BDBB  EF 81 37    rst 5,3781

----- INT ARITH, ??
      @ D83F! D851! D8CB! EE4D!
BDBE  EF 50 37    rst 5,3750

----- INT ARITH, DVDu <hl>=<hl>/<de>; <de>=remainder
      @ F2BF!
BDC1  EF 8C 37    rst 5,378C

----- INT ARITH, COMPARE <hl>,<de>; <a>= FF,00,01
      @ C6BE! FD0C
BDC4  EF E9 37    rst 5,37E9

----- INT ARITH, COMPLEMENT <hl>
      @ FD90! FE75!
BDC7  EF D4 37    rst 5,37D4

----- INT ARITH, get SGN of <hl>; <a>= FF,00,01
BDCA  EF E0 37    rst 5,37E0

----- TXT DRAW/UNDRAW CURSOR, if enabled
      @ 117D  1297  12BA  1347  13B8! 159D!
BDCD  C3 63 12    jp 1263

----- TXT DRAW/UNDRAW CURSOR, if enabled
      @ 10C2! 1177! 11A8! 128C! 129D! 12B2! 13AE! 1540!
BDD0  C3 63 12    jp 1263

----- TXT WRITE CHAR <a> on screen, <hl>=pos
      @ 1344!
BDD3  C3 4A 13    jp 134A

----- TXT UNWRITE CHAR, read screen <hl>=col/row, =<a>
      @ 13B4!
BDD6  C3 C0 13    jp 13C0

----- TXT OUT ACTION, char or ctl code <a> to VDU
      @ 1404!
BDD9  C3 0C 14    jp 140C
```

```
----- GRA PLOT a POINT, <de>=x, <hl>=y
      @ 1813
BDDC  C3 16 18     jp 1816

----- GRA TEST a POINT, <de>=x, <hl>=y
      @ 1827
BDDF  C3 2A 18     jp 182A

----- GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
      @ 1839
BDE2  C3 3C 18     jp 183C

----- SCR READ a pixel from the screen, (hl)=addr, <c>=mask
      @ 1833
BDE5  C3 82 0C     jp 0C82

----- SCR WRITE pixel(s) (hl)=addr, <c>=mask, using curr graph write mode
      @ 0FF7! 1008! 101F  1027  1041! 1821  19DC
BDE8  C3 68 0C     jp 0C68

----- SCR CLEAR screen to ink 0
      @ 0AE1!
BDEB  C3 F7 0A     jp 0AF7

----- KM TEST BREAK or reset; in: interrupts disabled, <c>=shft/ctl key states
      @ 1BEB!
BDEE  C3 2F 1C     jp 1C2F

----- MC WAIT PRINTER, print char <a> or time out
      @ 07F3!
BDF1  C3 F8 07     jp 07F8

----- not used so far
BDF4  00 00 00 00 00 00 00 00  00 00 00 00
_________________________________________________________________
BE00  E9 3F FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE18  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE30  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE48  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE60  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE78  FF FF FF FF FF FF FF FF  BF EF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BE90  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BEA8  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BEC0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BED8  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
BEF0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF

----- SYSTEM STACK
BF00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
BF18  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
BF30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
BF48  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
BF60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
BF78  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 20 20 20 00 00 00 00
BF90  00 00 00 00 00 00 00 00  20 20 20 20 00 00 00 F7  C0 1B D2 B9 10 04 41 05
BFA8  23 D7 14 AF 4B D6 D2 00  11 B9 EE 34 1D AF 1B 36  C2 B0 A9 2E D2 BD C2 B0
BFC0  00 AF 4B B9 00 40 11 35  EE 34 09 AF 1B 36 C2 F7  A9 2E 4F BD C2 B0 41 BA
BFD8  11 FE FE FD A7 D0 FA 01  FA 01 00 00 37 DA FA 01  D2 05 FF AE 02 02 1B D0
BFF0  07 8B 9A B9 8E 7F 03 8B  9A B9 86 7F EA F1 8B DD
```

```
----- on board ROM
      @ 0339> 05DC: BA86>
C000  80

----- ROM MARK#, VERSION#, REVISION LEVEL
      @ BA89>
C001  01 00 00

----- EXTERNAL COMMAND TABLE
      @ 02F4:
C004  4C C0       C04C            'Basic   (command table, only one entry)

----- entry to upper ROM
      @ 007D: 0342!
C006  31 00 C0    ld sp,C000      init stack pointer
C009  CD CB BC    call BCCB       KL ROM WALK, (de)=low, (hl)=hi avail. memory
C00C  CD C4 F4    call F4C4       init all Basic pointers
C00F  DA 00 00    jp c,0000       SYSTEM RESET if carry
C012  21 00 AC    ld hl,AC00      BASIC flag ??
C015  36 00       ld (hl),00      reset to 0
C017  06 1B       ld b,1B         count of indirections *3
C019  23          inc hl          reset all indirections to return
C01A  36 C9       ld (hl),C9      = ret
C01C  10 FB       djnz C019       next
C01E  21 3F C0    ld hl,C03F      'Basic 1.0
C021  CD 37 C3    call C337       set default printer WIDTH 132.
C024  AF          xor a           =0
C025  32 00 AC    ld (AC00),a     BASIC flag ??
C028  CD CB DD    call DDCB       reset BASIC program counter
C02B  CD 84 CA    call CA84       reset last error# to 0
C02E  CD 97 BD    call BD97       REAL ARITH, set initial RANDOM NUMBER
C031  CD D3 C0    call C0D3       reset flag for AUTO
C034  CD 3E C1    call C13E       performs command NEW
C037  11 F0 00    ld de,00F0      default VAL for SYMBOL AFTER =240.
C03A  CD 06 F7    call F706       set SYMBOL AFTER <de>
C03D  18 25       jr C064         reset Basic

----- 'Basic 1.0
C03F  20 42 41 53 49 43 20 31  2E 30 0A 0A 00               ' BASIC 1.0...

----- 'Basic   (command table, only one entry)
C04C  42 41 53 49 C3 00                                     'BASIC.

----- command: EDIT <line#>
      @ DE2D
C052  CD E1 CE    call CEE1       get line# into <de>
C055  C0          ret nz

----- perform EDIT line# <de>
      @ C08E'
C056  31 00 C0    ld sp,C000      init stack pointer
C059  CD 9A E7    call E79A       search line# <de> from start, <hl>=addr, nc=
C05C  CD 63 E1    call E163       list a basic line into the edit buffer
C05F  CD 43 CA    call CA43       keyboard edit line in edit buffer
C062  38 54       jr c,C0B8       check line for direct command

----- reset Basic
      @ C03D' C0A2' C12F  CADC  CB90  E10A  E734  E861  E9FE  EAB2
C064  CD 01 AC    call AC01       Indirection: RESET Basic
C067  31 00 C0    ld sp,C000      init stack pointer
C06A  CD 62 C1    call C162       reset string stack, FN pointers, I/O chan=0
C06D  CD D6 DD    call DDD6       get BASIC line# at PC in <hl>, =carry
C070  DC B6 BC    call c,BCB6     SOUND HOLD, stop all sounds
C073  CD 48 BB    call BB48       KM DISARM BREAK
C076  CD 86 C3    call C386       set chan 0, print char <a> at a new line
```

```
C079  3A 45 AE    ld a,(AE45)      flag file read protected
C07C  B7          or a
C07D  C4 3E C1    call nz,C13E     performs command NEW
C080  3A AA AD    ld a,(ADAA)      last Basic ERROR number
C083  D6 02       sub 02           Syntax error?
C085  20 09       jr nz,C090       no, not a Syntax error
C087  32 AA AD    ld (ADAA),a      last Basic ERROR number
C08A  CD DF CA    call CADF        get next line after error <hl>, if none <hl>
C08D  EB          ex de,hl
C08E  38 C6       jr c,C056        perform EDIT line# <de>
C090  21 CC C0    ld hl,C0CC       'Ready
C093  CD 41 C3    call C341        output text (hl) to channel

----- reset BASIC pointer; test AUTO; wait for input
      @ C0C0' C0FF
C096  CD CB DD    call DDCB        reset BASIC program counter

----- test AUTO flag; (print line#); wait for input
C099  3A 1C AC    ld a,(AC1C)      flag for AUTO
C09C  B7          or a
C09D  28 11       jr z,C0B0        get a command line from keyboard
C09F  CD 02 C1    call C102        get new line#
C0A2  30 C0       jr nc,C064       reset Basic
C0A4  7E          ld a,(hl)
C0A5  B7          or a
C0A6  28 F1       jr z,C099        test AUTO flag; (print line#); wait for inpu
C0A8  CD D2 E6    call E6D2        assemble an insert line into program
C0AB  CD 7A C1    call C17A        reset pointers; program, error, data
C0AE  18 E9       jr C099          test AUTO flag; (print line#); wait for inpu

----- get a command line from keyboard
C0B0  CD 3B CA    call CA3B        put 0 in edit buffer and read a line
C0B3  30 FB       jr nc,C0B0       get a command line from keyboard
C0B5  CD 4E C3    call C34E        output 'LF to channel

----- check line for direct command
C0B8  CD BC E6    call E6BC        check for a line# in command line
C0BB  30 05       jr nc,C0C2       perform a direct command
C0BD  C4 7A C1    call nz,C17A     reset pointers; program, error, data
C0C0  18 D4       jr C096          reset BASIC pointer; test AUTO; wait for inp

----- perform a direct command
C0C2  CD BB DE    call DEBB        assemble a program line; (hl)=edit buffer
C0C5  CD 53 C4    call C453        establish BREAK EVENT
C0C8  2B          dec hl
C0C9  C3 74 DD    jp DD74          do the RUN LOOP

----- 'Ready
C0CC  52 65 61 64 79 0A 00                                  'Ready..

----- reset flag for AUTO
      @ C031! C116! C16E!
C0D3  AF          xor a            a=0
C0D4  18 05       jr C0DB          reset flag for AUTO

----- set flag for AUTO
      @ C0FB! C125!
C0D6  22 1D AC    ld (AC1D),hl     new line number
C0D9  3E FF       ld a,FF          set
C0DB  32 1C AC    ld (AC1C),a      flag for AUTO
C0DE  C9          ret
```

```
----- command: AUTO [<line#>][,<line step>]
      @ DE03
C0DF  11 0A 00     ld de,000A        default START = 10.
C0E2  28 02        jr z,C0E6
C0E4  FE 2C        cp 2C             ',
C0E6  C4 E1 CE     call nz,CEE1      get line# into <de>
C0E9  D5           push de
C0EA  11 0A 00      ld de,000A       default STEP = 10.
C0ED  CD 55 DD      call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
C0F0  DC E1 CE      call c,CEE1      get line# into <de>
C0F3  CD 4A DD      call DD4A        CHRGOT <a>; end of statement? else syntax er
C0F6  EB            ex de,hl
C0F7  22 1F AC      ld (AC1F),hl     step for AUTO
C0FA  E1           pop hl
C0FB  CD D6 C0     call C0D6         set flag for AUTO
C0FE  C1           pop bc
C0FF  C3 96 C0     jp C096           reset BASIC pointer; test AUTO; wait for inp

----- get new line#
      @ C09F!
C102  2A 1D AC     ld hl,(AC1D)      new line number
C105  E5           push hl
C106  CD 79 EE      call EE79        print line#
C109  D1           pop de
C10A  CD A3 E7     call E7A3         search line# <de> from start, <hl>=address,
C10D  3E 2A        ld a,2A           '* = warning,
C10F  38 02        jr c,C113          line exists!
C111  3E 20        ld a,20           'SPACE
C113  CD 56 C3     call C356         output char <a> to channel
C116  CD D3 C0     call C0D3         reset flag for AUTO
C119  CD 3B CA     call CA3B         put 0 in edit buffer and read a line
C11C  D0           ret nc            break
C11D  CD 4E C3     call C34E         output 'LF to channel
C120  E5           push hl
C121  2A 1F AC      ld hl,(AC1F)     step for AUTO
C124  19            add hl,de
C125  D4 D6 C0      call nc,C0D6     set flag for AUTO
C128  E1           pop hl
C129  37           scf
C12A  C9           ret

----- command: NEW
      @ DE63
C12B  C0           ret nz
C12C  CD 3E C1     call C13E         performs command NEW
C12F  C3 64 C0     jp C064           reset Basic

----- command: CLEAR
      @ DE0D
C132  E5           push hl
C133  CD 8C C1      call C18C        reset all VARIABLE pointers
C136  CD 5B C1      call C15B        cas I/O abandon, set RAD, release buffers
C139  CD 7A C1      call C17A        reset pointers; program, error, data
C13C  E1           pop hl
C13D  C9           ret

----- performs command NEW
      @ C034! C07D! C12C!
C13E  2A 7F AE     ld hl,(AE7F)      low memory boundary pointer
C141  EB           ex de,hl
C142  2A 7B AE     ld hl,(AE7B)      himem for Basic pointer
C145  CD DA FF     call FFDA         BC=HL-DE
C148  62           ld h,d
C149  6B           ld l,e            hl=de
C14A  13           inc de
```

```
C14B  AF          xor a
C14C  77          ld (hl),a         clear to 0
C14D  ED B0       ldir
C14F  32 45 AE    ld (AE45),a       flag file read protected
C152  CD 76 E6    call E676         reset line MARK, basic end=start
C155  CD 8C C1    call C18C         reset all VARIABLE pointers
C158  CD 6B C1    call C16B         performs NEW, part 2

----- cas I/O abandon, set RAD, release buffers
      @ C136!
C15B  CD AD D2    call D2AD         CAS in/out abandon, release I/O buffers

----- reset to RAD
      @ E9ED!
C15E  AF          xor a             =0 =RAD
C15F  CD 73 BD    call BD73         REAL ARITH, set DEG/RAD <a>

----- reset string stack, FN pointers, I/O chan=0
      @ C06A!
C162  CD B3 FB    call FBB3         reset string stack
C165  CD FD D9    call D9FD         reset FN pointers as not used
C168  C3 9D C1    jp C19D           set in/out channel to 0

----- performs NEW, part 2
      @ C158! EA1C! EA8A! EB41!
C16B  CD E6 DD    call DDE6         command: TROFF
C16E  CD D3 C0    call C0D3         reset flag for AUTO
C171  CD F2 F1    call F1F2         set default ZONE to 13.
C174  CD 76 E6    call E676         reset line MARK, basic end=start
C177  CD B1 D5    call D5B1         reset all VARIABLE pointers to basic start

----- reset pointers; program, error, data
      @ C0AB! C0BD! C139! E731! E9EA! EAB5! EBB8! EBEF!
C17A  CD D9 CB    call CBD9         reset ON ERROR FLAG and ADDRESS
C17D  CD AB CB    call CBAB         reset CONTinue pointer
C180  CD ED C8    call C8ED         set up all default events
C183  CD 8E F5    call F58E         reset BASIC STACK
C186  CD D2 D5    call D5D2         clear AE04..5 to 0
C189  C3 E5 DC    jp DCE5           reset DATA pointer to basic start

----- reset all VARIABLE pointers
      @ C133! C155! E9E7! EA19! EAAC! EB3E!
C18C  C5          push bc
C18D  E5           push hl
C18E  CD CA F5      call F5CA       set low string end up to himem
C191  CD AE D5      call D5AE       reset all VARIABLE pointers to Basic start,
C194  CD FC D5      call D5FC       set default VARTYPE A-Z to real
C197  CD 89 E9      call E989       clear all VARIABLE indices
C19A  E1           pop hl
C19B  C1          pop bc
C19C  C9          ret

----- set in/out channel to 0
      @ C168  C33D! CB36!
C19D  AF          xor a             =0
C19E  CD AF C1    call C1AF         set input chan to <a>, a=old channel
C1A1  AF          xor a             =0

----- set output channel to <a>, a= old channel
      @ C1C9' C1D7! C1E0  C27C! C28D  CB06! CB11! DB20! DB28  DB4C! DB74
      @ EC8D! EC9B! F205  F4C1
C1A2  E5          push hl
C1A3  F5           push af
C1A4  FE 08         cp 08           is it the printer?
C1A6  DC B4 BB      call c,BBB4     TXT STREAM <a> SELECT, <a>=old text stream
```

```
C1A9  F1          pop af
C1AA  21 21 AC    ld hl,AC21      output channel number
C1AD  18 04       jr C1B3

----- set input chan to <a>, a=old channel
      @ C19E! C1CE' DB17  DB44
C1AF  E5          push hl
C1B0  21 22 AC     ld hl,AC22     input channel number
C1B3  D5           push de
C1B4  5F            ld e,a
C1B5  7E            ld a,(hl)
C1B6  73            ld (hl),e
C1B7  D1           pop de
C1B8  E1          pop hl
C1B9  C9          ret

----- get output channel; cp 08
      @ E145!
C1BA  3A 21 AC    ld a,(AC21)     output channel number
C1BD  FE 08       cp 08           is it the printer?
C1BF  C9          ret

----- get input channel; cp 09
      @ DB1A! DB47! DB98! DC10! DC21!
C1C0  3A 22 AC    ld a,(AC22)     input channel number
C1C3  FE 09       cp 09           is it the tape?
C1C5  C9          ret

----- if '# get chan; default=0; set output channel
      @ E0FC! F1FD! F47B!
C1C6  CD E3 C1    call C1E3       if '# get channel, default=0
C1C9  18 D7       jr C1A2         set output channel to <a>, a= old channel

----- if '# get chan, default=0, set input channel
      @ DAFC! DB2B!
C1CB  CD E3 C1    call C1E3       if '# get channel, default=0
C1CE  18 DF       jr C1AF         set input chan to <a>, a=old channel

----- get channel#, default=0; set in/out chan
      @ C20A! C212! C25A! C2D2! C2E6! C319! C320!
C1D0  CD E3 C1    call C1E3       if '# get channel, default=0
C1D3  FE 08       cp 08           is channel >= 8?
C1D5  30 2E       jr nc,C205      Error: Improper argument
C1D7  CD A2 C1    call C1A2       set output channel to <a>, a= old channel
C1DA  C1          pop bc
C1DB  F5          push af
C1DC  CD F9 FF     call FFF9      jp(bc)
C1DF  F1          pop af
C1E0  C3 A2 C1    jp C1A2         set output channel to <a>, a= old channel

----- if '# get channel, default=0
      @ C1C6! C1CB! C1D0!
C1E3  7E          ld a,(hl)
C1E4  FE 23       cp 23           '#
C1E6  3E 00       ld a,00         default 0, if '# is missing
C1E8  C0          ret nz
C1E9  CD F5 C1    call C1F5       get next VAL in <a>; max=10.; else error
C1EC  F5          push af
C1ED  CD 55 DD     call DD55      CHRBACK comma?; if=:CHRGET <a>, scf
C1F0  D4 4A DD     call nc,DD4A   CHRGOT <a>; end of statement? else syntax er
C1F3  F1          pop af
C1F4  C9          ret
```

```
----- get next VAL in <a>; max=10.; else error
      @ C1E9! C279!
C1F5  CD 37 DD      call DD37         CHRNEXT <a>, nz=Error; CHRGET
C1F8  23            '#
C1F9  3E 0A         ld a,0A           maximum

----- get next VAL in <a>; cp <old a>; nc=error
      @ C246! C24D' C251! C314!
C1FB  C5            push bc
C1FC  D5             push de
C1FD  47              ld b,a
C1FE  CD 67 CE        call CE67       get byte VAL(expression) in <de>
C201  B8              cp b
C202  D1             pop de
C203  C1            pop bc
C204  D8            ret c

----- Error: Improper argument
C205  1E 05         ld e,05           Improper argument
C207  C3 94 CA      jp CA94           perform ERROR <e> routine

----- command: PAPER [#<device>,] <ink>
      @ DE75
C20A  CD D0 C1      call C1D0         get channel#, default=0; set in/out chan
C20D  01 96 BB      ld bc,BB96        TXT SET PAPER ink <a>
C210  18 06         jr C218

----- command: PEN [#<device>,] <ink>
      @ DE77
C212  CD D0 C1      call C1D0         get channel#, default=0; set in/out chan
C215  01 90 BB      ld bc,BB90        TXT SET PEN ink <a>
C218  CD 4B C2      call C24B         get VAL into <a>, max=15.; else error
C21B  E5            push hl
C21C  CD F9 FF       call FFF9        jp(bc)
C21F  E1            pop hl
C220  C9            ret

----- command: BORDER <ink> [,<ink>]
      @ DE05
C221  CD 3C C2      call C23C         get VAL in <b>, next in <a>, both max 31.
C224  E5            push hl
C225  CD 38 BC       call BC38        SCR SET BORDER, <b,c>=colours
C228  E1            pop hl
C229  C9            ret

----- command: INK<ink>,<colour>[,<colour>]
      @ DE45
C22A  CD 4B C2      call C24B         get VAL into <a>, max=15.; else error
C22D  F5            push af
C22E  CD 37 DD       call DD37        CHRNEXT <a>, nz=Error; CHRGET
C231  2C             ',
C232  CD 3C C2       call C23C        get VAL in <b>, next in <a>, both max 31.
C235  F1            pop af
C236  E5            push hl
C237  CD 32 BC       call BC32        SCR SET colour of INK, <a>=ink#, <b,c>=colou
C23A  E1            pop hl
C23B  C9            ret

----- get VAL in <b>, next in <a>, both max 31.
      @ C221! C232!
C23C  CD 44 C2      call C244         get VAL into <a>, max=31.; else error
C23F  41            ld b,c
C240  CD 55 DD      call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
C243  D0            ret nc
```

```
----- get VAL into <a>, max=31.; else error
      @ C23C!
C244  3E 20       ld a,20          =32.
C246  CD FB C1    call C1FB        get next VAL in <a>; cp <old a>; nc=error
C249  4F          ld c,a
C24A  C9          ret

----- get VAL into <a>, max=15.; else error
      @ C218! C22A! C4BA! C4E1!
C24B  3E 10       ld a,10          =16.
C24D  18 AC       jr C1FB          get next VAL in <a>; cp <old a>; nc=error

----- command: MODE <mode>
      @ DE5B
C24F  3E 03       ld a,03          =3
C251  CD FB C1    call C1FB        get next VAL in <a>; cp <old a>; nc=error
C254  E5          push hl
C255  CD 0E BC     call BC0E       SCR SET MODE <a>
C258  E1          pop hl
C259  C9          ret

----- command: CLS [#<device>]
      @ DE15
C25A  CD D0 C1    call C1D0        get channel#, default=0; set in/out chan
C25D  3E 0C       ld a,0C          'FF  (^L)
C25F  C3 6E C3    jp C36E          output char <a> to channel

----- function: VPOS(#<device>)
      @ D1AC
C262  01 67 C2    ld bc,C267       get VPOS of <device>
C265  18 12       jr C279

----- get VPOS of <device>
      @ C262:
C267  3A 21 AC    ld a,(AC21)      output channel number
C26A  FE 08       cp 08            is it >= 8?
C26C  30 97       jr nc,C205       Error: Improper argument
C26E  CD 78 BB    call BB78        TXT GET CURSOR position (hl), roll count <a>
C271  CD 87 BB    call BB87        TXT VALIDATE cursor position <hl> column/row
C274  7D          ld a,l
C275  C9          ret

----- function: POS(#<device>)
      @ D19E
C276  01 90 C2    ld bc,C290       get current print POS of <device>
C279  CD F5 C1    call C1F5        get next VAL in <a>; max=10.; else error
C27C  CD A2 C1    call C1A2        set output channel to <a>, a= old channel
C27F  F5          push af
C280  CD 37 DD     call DD37       CHRNEXT <a>, nz=Error; CHRGET
C283  29           ')
C284  E5           push hl
C285  CD F9 FF      call FFF9      jp(bc)
C288  CD 0A FF      call FF0A      set FAC to <a> and mark integer
C28B  E1           pop hl
C28C  F1          pop af
C28D  C3 A2 C1    jp C1A2          set output channel to <a>, a= old channel

----- get current print POS of <device>
      @ C276: C2C6! F263! F287!
C290  3A 21 AC    ld a,(AC21)      output channel number
C293  FE 08       cp 08            is it the printer?
C295  CA DF C3    jp z,C3DF        get POS (printer) <a>
C298  3A 25 AC    ld a,(AC25)      POS(tape); # of char's written on this line
C29B  D0          ret nc
C29C  C3 9C C3    jp C39C          get cursor position and validate
```

```
----- get line width of <device>
      @ C2C0! F2B6!
C29F  3A 21 AC     ld a,(AC21)       output channel number
C2A2  FE 08        cp 08
C2A4  28 0D        jr z,C2B3         is it the printer?
C2A6  D0           ret nc            always return if printer
C2A7  D5           push de
C2A8  E5            push hl
C2A9  CD 69 BB       call BB69       TXT GET WINDOW size, <hl>=left top, <de>=rig
C2AC  7A             ld a,d
C2AD  94             sub h
C2AE  3C             inc a
C2AF  E1            pop hl
C2B0  D1           pop de
C2B1  37           scf
C2B2  C9           ret

----- get WIDTH; cp FF
C2B3  3A 24 AC     ld a,(AC24)       WIDTH for Printer
C2B6  FE FF        cp FF
C2B8  C9           ret

----- check if char fits into this line
      @ F24D! F26E!
C2B9  E5           push hl
C2BA  CD BF C2      call C2BF        do it here!
C2BD  E1           pop hl
C2BE  C9           ret

----- do it here!
      @ C2BA!
C2BF  67           ld h,a
C2C0  CD 9F C2     call C29F         get line width of <device>
C2C3  3F           ccf
C2C4  D8           ret c
C2C5  6F           ld l,a
C2C6  CD 90 C2     call C290         get current print POS of <device>
C2C9  3D           dec a
C2CA  37           scf
C2CB  C8           ret z
C2CC  84           add a,h
C2CD  3F           ccf
C2CE  D0           ret nc
C2CF  3D           dec a
C2D0  BD           cp l
C2D1  C9           ret

----- command: LOCATE [#<device>,] <x coord>, <y coord>
      @ DE53
C2D2  CD D0 C1     call C1D0         get channel#, default=0; set in/out chan
C2D5  CD 27 C3     call C327         <d>=next VAL-1, <e>=next VAL-1
C2D8  E5           push hl
C2D9  EB            ex de,hl
C2DA  24            inc h
C2DB  2C            inc l
C2DC  CD 75 BB      call BB75        TXT SET CURSOR, <hl>=column/row
C2DF  E1           pop hl
C2E0  C9           ret

----- command: WINDOW [#<device>,]<left>,<right>,<top>,<bottom>
      @ DEB1
C2E1  7E           ld a,(hl)
C2E2  FE E7        cp E7             [SWAP]
C2E4  28 17        jr z,C2FD         command: WINDOW SWAP [<device>,] <device>
C2E6  CD D0 C1     call C1D0         get channel#, default=0; set in/out chan
```

```
C2E9  CD 27 C3    call C327         <d>=next VAL-1, <e>=next VAL-1
C2EC  D5          push de
C2ED  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
C2F0  2C           ',
C2F1  CD 27 C3     call C327        <d>=next VAL-1, <e>=next VAL-1
C2F4  E3           ex (sp),hl
C2F5  7A           ld a,d
C2F6  55           ld d,l
C2F7  6F           ld l,a
C2F8  CD 66 BB     call BB66        TXT SET WINDOW <hl>=left top, <de>=right bot
C2FB  E1          pop hl
C2FC  C9          ret

----- command: WINDOW SWAP [<device>,] <device>
C2FD  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
C300  CD 12 C3    call C312         get byte VAL into <b>; max=7; else error
C303  48          ld c,b
C304  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
C307  06 00       ld b,00           default=0
C309  DC 12 C3    call c,C312       second argument
C30C  E5          push hl
C30D  CD B7 BB     call BBB7        TXT SWAP STREAMS <b> with <c>
C310  E1          pop hl
C311  C9          ret

----- get byte VAL into <b>; max=7; else error
      @ C300! C309!
C312  3E 08       ld a,08           max+1
C314  CD FB C1    call C1FB         get next VAL in <a>; cp <old a>; nc=error
C317  47          ld b,a
C318  C9          ret

----- command: TAG [#<device>]
      @ DEA1
C319  CD D0 C1    call C1D0         get channel#, default=0; set in/out chan
C31C  3E FF       ld a,FF           flag ON
C31E  18 04       jr C324           set graphic char write

----- command: TAGOFF [#<device>]
      @ DEA3
C320  CD D0 C1    call C1D0         get channel#, default=0; set in/out chan
C323  AF          xor a             flag OFF
C324  C3 63 BB    jp BB63           TXT SET GRAPHIC char write, <a>=0=OFF, FF=ON

----- <d>=next VAL-1, <e>=next VAL-1
      @ C2D5! C2E9! C2F1!
C327  CD 2F C3    call C32F         <a>=next VAL, 0=error, <e>=<a>-1
C32A  53          ld d,e
C32B  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
C32E  2C          ',

----- <a>=next VAL, 0=error, <e>=<a>-1
      @ C327!
C32F  D5          push de
C330  CD 6D CE     call CE6D        <a>=next VAL, 0=error
C333  D1          pop de
C334  5F          ld e,a
C335  1D          dec e
C336  C9          ret

----- set default printer WIDTH 132.
      @ C021!
C337  3E 84       ld a,84           =132.
C339  32 24 AC    ld (AC24),a       WIDTH for Printer
C33C  E5          push hl
```

```
C33D  CD 9D C1      call C19D       set in/out channel to 0
C340  E1           pop hl

----- output text (hl) to channel
      @ C093! CB0A! CB3D! CB48! D565! DB6C! EE7F  F3B3!
C341  F5           push af
C342  E5            push hl
C343  7E             ld a,(hl)
C344  23             inc hl
C345  B7             or a
C346  C4 56 C3       call nz,C356   output char <a> to channel
C349  20 F8          jr nz,C343     next char
C34B  E1            pop hl
C34C  F1           pop af
C34D  C9           ret

----- output 'LF to channel
      @ C0B5! C11D! CA49  CB0D! D56E! DBB8! E13B! F20B  F250! F258! F271
      @ F290! F31B! F4BD!
C34E  F5           push af
C34F  3E 0A         ld a,0A         'LF  (^J)
C351  CD 56 C3      call C356       output char <a> to channel
C354  F1           pop af
C355  C9           ret

----- output char <a> to channel
      @ C113! C346! C351! DB56! DB5B! DDED! DDFE  F29B! F33C! F49B! F4A3!
      @ F4B8!
C356  F5           push af
C357  CD 5C C3      call C35C       perform output char <a> to <device>
C35A  F1           pop af
C35B  C9           ret

----- perform output char <a> to <device>
      @ C357!
C35C  FE 0A        cp 0A            is it a 'LF ?
C35E  20 0E        jr nz,C36E       output char <a> to channel
C360  3A 21 AC     ld a,(AC21)      output channel number
C363  FE 08        cp 08            is it the printer?
C365  CA A8 C3     jp z,C3A8        output 'CR 'LF to printer
C368  D2 EA C3     jp nc,C3EA       output 'CR 'LF to tape
C36B  C3 92 C3     jp C392          output 'CR 'LF to screen

----- output char <a> to channel
      @ C25F  C35E' E14B! E15C! E160  F82E!
C36E  F5           push af
C36F  C5            push bc
C370  4F             ld c,a
C371  CD 77 C3       call C377      do it here!
C374  C1            pop bc
C375  F1           pop af
C376  C9           ret

----- do it here!
      @ C371!
C377  3A 21 AC     ld a,(AC21)      output channel number
C37A  FE 08        cp 08            is it the printer?
C37C  CA B5 C3     jp z,C3B5        output char <c> to printer; update POS
C37F  D2 F8 C3     jp nc,C3F8       output char <a> to tape; update POS
C382  79           ld a,c
C383  C3 99 C3     jp C399          TXT OUTPUT char or ctl code <a> to VDU
```

```
----- set chan 0, print char <a> at a new line
      @ C076! CB18! CB39!
C386  AF          xor a
C387  CD 63 BB    call BB63        TXT SET GRAPHIC char write, <a>=0=OFF, FF=ON
C38A  CD 54 BB    call BB54        TXT VDU ENABLE
C38D  CD 9C C3    call C39C        get cursor position and validate
C390  3D          dec a
C391  C8          ret z

----- output 'CR 'LF to screen
C392  3E 0D       ld a,0D          'CR  (^M)
C394  CD 99 C3    call C399        TXT OUTPUT char or ctl code <a> to VDU
C397  3E 0A       ld a,0A          'LF  (^J)

----- TXT OUTPUT char or ctl code <a> to VDU
      @ C383  C394!
C399  C3 5A BB    jp BB5A          TXT OUTPUT char or ctl code <a> to VDU

----- get cursor position and validate
      @ C29C  C38D!
C39C  C5          push bc
C39D  E5           push hl
C39E  CD 78 BB      call BB78      TXT GET CURSOR position (hl), roll count <a>
C3A1  CD 87 BB      call BB87      TXT VALIDATE cursor position <hl> column/row
C3A4  7C            ld a,h
C3A5  E1           pop hl
C3A6  C1          pop bc
C3A7  C9          ret

----- output 'CR 'LF to printer
      @ C365  C3C8!
C3A8  C5          push bc
C3A9  0E 0D        ld c,0D         'CR  (^M)
C3AB  CD B5 C3     call C3B5       output char <c> to printer; update POS
C3AE  0E 0A        ld c,0A         'LF  (^J)
C3B0  CD B5 C3     call C3B5       output char <c> to printer; update POS
C3B3  C1          pop bc
C3B4  C9          ret

----- output char <c> to printer; update POS
      @ C37C  C3AB! C3B0!
C3B5  E5          push hl
C3B6  79           ld a,c
C3B7  EE 0D        xor 0D          'CR  (^M)
C3B9  28 13        jr z,C3CE       yes, start new char count
C3BB  79           ld a,c
C3BC  FE 20        cp 20           'SPACE
C3BE  38 14        jr c,C3D4       not a printable character
C3C0  2A 23 AC     ld hl,(AC23)    POS(printer); # of char's written this line
C3C3  24           inc h
C3C4  7D           ld a,l          = WIDTH
C3C5  28 07        jr z,C3CE       not beyond
C3C7  BC           cp h            width reached?
C3C8  CC A8 C3     call z,C3A8     output 'CR 'LF to printer
C3CB  3A 23 AC     ld a,(AC23)     POS(printer); # of char's written this line
C3CE  3C           inc a           inc char count
C3CF  28 03        jr z,C3D4       skip if zero
C3D1  32 23 AC     ld (AC23),a     update char count this line
C3D4  E1          pop hl
C3D5  79          ld a,c
C3D6  CD 2B BD    call BD2B        MC PRINT CHAR <a> to Centronics port
C3D9  D8          ret c
C3DA  CD 3C C4    call C43C        check for a BREAK request
C3DD  18 F6       jr C3D5          next
```

```
----- get POS (printer) <a>
      @ C295
C3DF  3A 23 AC    ld a,(AC23)      POS(printer); # of char's written this line
C3E2  C9          ret

----- command: WIDTH <width>
      @ DEAF
C3E3  CD 6D CE    call CE6D        <a>=next VAL, 0=error
C3E6  32 24 AC    ld (AC24),a      WIDTH for Printer
C3E9  C9          ret

----- output 'CR 'LF to tape
      @ C368
C3EA  3E 01       ld a,01          char count=1
C3EC  32 25 AC    ld (AC25),a      POS(tape); # of char's written on this line
C3EF  3E 0D       ld a,0D          'CR  (^M)
C3F1  CD 0D C4    call C40D        output char <a> to tape
C3F4  3E 0A       ld a,0A          'LF  (^J)
C3F6  18 15       jr C40D          output char <a> to tape

----- output char <a> to tape; update POS
      @ C37F
C3F8  E5          push hl
C3F9  21 25 AC     ld hl,AC25      POS(tape); # of char's written on this line
C3FC  79           ld a,c
C3FD  06 01        ld b,01         new count =1
C3FF  FE 0D        cp 0D           is it a 'CR?
C401  28 08        jr z,C40B       yes!
C403  FE 20        cp 20           'SPACE
C405  38 05        jr c,C40C       not a printable char
C407  46           ld b,(hl)
C408  04           inc b           inc count
C409  28 01        jr z,C40C       skip if zero
C40B  70           ld (hl),b       update char count this line
C40C  E1          pop hl

----- output char <a> to tape
      @ C3F1! C3F6'
C40D  CD 95 BC    call BC95        CAS OUT CHAR <a> to output file
C410  D8          ret c

----- perform a BREAK
C411  C3 6B CB    jp CB6B          perform a BREAK

----- CAS RETURN, put last char read back
      @ DC99! DCBF!
C414  C3 86 BC    jp BC86          CAS RETURN, put last char read back

----- function: EOF
      @ D0CA:
C417  E5          push hl
C418  CD 89 BC     call BC89       CAS TEST EOF
C41B  28 F4        jr z,C411       perform a BREAK
C41D  3F           ccf
C41E  9F           sbc a,a
C41F  CD 05 FF     call FF05       set FAC to (-1, 0, +1); <a> was FF,00,01
C422  E1          pop hl
C423  C9          ret

----- read a char from input file
      @ DCA8! DCB9!
C424  3A 22 AC    ld a,(AC22)      input channel number
C427  FE 09       cp 09            is it the tape?
C429  CA 80 BC    jp z,BC80        CAS IN CHAR from input file
C42C  CD 09 BB    call BB09        KM READ CHAR from keyboard =<a>
```

```
C42F  D8          ret c

----- cursor ON; wait for key; cursor OFF
      @ C473!
C430  CD 81 BB    call BB81         TXT CURSOR ON
C433  CD 06 BB    call BB06         KM WAIT CHAR from keyboard =<a>
C436  C3 84 BB    jp BB84           TXT CURSOR OFF

----- jp KM read char from keyboard
      @ FA2A!
C439  C3 09 BB    jp BB09           KM READ CHAR from keyboard =<a>

----- check for a BREAK request
      @ C3DA! E11B!
C43C  CD 09 BB    call BB09         KM READ CHAR from keyboard =<a>
C43F  D0          ret nc            no key, return
C440  FE FC       cp FC             is it 'ESC?
C442  C0          ret nz            no, return
C443  C5          push bc
C444  D5           push de
C445  E5            push hl
C446  CD 6F C4       call C46F      check for a second 'ESC
C449  DA 6B CB       jp c,CB6B      perform a BREAK
C44C  CD 53 C4       call C453      establish BREAK EVENT
C44F  E1            pop hl
C450  D1           pop de
C451  C1          pop bc
C452  C9          ret

----- establish BREAK EVENT
      @ C0C5! C44C! C832! C8C1! C90C!
C453  E5          push hl
C454  11 5E C4     ld de,C45E       event routine BREAK
C457  0E FD        ld c,FD          select upper on board ROM
C459  CD 45 BB     call BB45        KM ARM BREAK, (de)=routine, <c>=ROM select
C45C  E1          pop hl
C45D  C9          ret

----- event routine BREAK
      @ B511: C454:
C45E  E5          push hl
C45F  CD 09 BB     call BB09        KM READ CHAR from keyboard =<a>
C462  30 04        jr nc,C468       all chars read?
C464  FE EF        cp EF            'ESC marker found?
C466  20 F7        jr nz,C45F       no, not yet
C468  CD 6F C4     call C46F        check for a second 'ESC
C46B  E1          pop hl
C46C  C3 47 C8    jp C847           event routine BREAK, part 2

----- check for a second 'ESC
      @ C446! C468!
C46F  CD B6 BC    call BCB6         SOUND HOLD, stop all sounds
C472  F5          push af
C473  CD 30 C4     call C430        cursor ON; wait for key; cursor OFF
C476  FE EF        cp EF            is it 'ESC marker?
C478  28 F9        jr z,C473        yes, get next key
C47A  FE FC        cp FC            is it ESC?
C47C  28 0B        jr z,C489        yes, return with carry set
C47E  FE 20        cp 20            'SPACE
C480  C4 0C BB     call nz,BB0C     KM RETURN CHAR <a> to 'put back' location
C483  F1          pop af
C484  DC B9 BC    call c,BCB9       SOUND CONTINUE stopped sounds
C487  B7          or a
C488  C9          ret
```

```
C489  F1          pop af
C48A  37          scf
C48B  C9          ret

----- command: ORIGIN <x>,<y> [,<left>,<right>,<top>,<bottom>]
      @ DE71
C48C  CD 1A C5    call C51A         get integer VAL to <de>, next to <bc>
C48F  C5          push bc
C490  D5           push de
C491  CD 55 DD      call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
C494  30 18         jr nc,C4AE      no window arguments
C496  CD 1A C5      call C51A       get integer VAL to <de>, next to <bc>
C499  C5            push bc
C49A  D5             push de
C49B  CD 37 DD        call DD37     CHRNEXT <a>, nz=Error; CHRGET
C49E  2C              inc l
C49F  CD 1A C5        call C51A     get integer VAL to <de>, next to <bc>
C4A2  C5              push bc
C4A3  E3               ex (sp),hl
C4A4  CD D2 BB         call BBD2    GRA set WINDOW height, <de>=y1, <hl>=y2
C4A7  E1              pop hl
C4A8  D1             pop de
C4A9  E3             ex (sp),hl
C4AA  CD CF BB       call BBCF      GRA set WINDOW width, <de>=x1, <hl>=x2
C4AD  E1            pop hl
C4AE  D1           pop de
C4AF  E3           ex (sp),hl
C4B0  CD C9 BB      call BBC9       GRA SET ORIGIN, <de>=x, <hl>=y
C4B3  E1          pop hl
C4B4  C9          ret

----- command: CLG [<ink>]
      @ DE0F
C4B5  CD 51 DD    call DD51         CHRGOT <a>; end of statement? =carry
C4B8  38 06       jr c,C4C0         no argument <ink>
C4BA  CD 4B C2    call C24B         get VAL into <a>, max=15.; else error
C4BD  CD E4 BB    call BBE4         GRA SET PAPER, <a>=ink
C4C0  E5          push hl
C4C1  CD DB BB     call BBDB        GRA CLEAR GRAPHIC WINDOW
C4C4  E1          pop hl
C4C5  C9          ret

----- command: DRAW <x>,<y>[,<ink>]
      @ DE29
C4C6  01 F6 BB    ld bc,BBF6        GRA DRAW LINE ABSOLUTE, <de>=x, <hl>=y
C4C9  18 0D       jr C4D8

----- command: DRAWR <xd>,<yd>[,<ink>]
      @ DE2B
C4CB  01 F9 BB    ld bc,BBF9        GRA DRAW LINE RELATIVE, <de>=xd, <hl>=yd
C4CE  18 08       jr C4D8

----- command: PLOT <x>,<y>[,<ink>]
      @ DE79
C4D0  01 EA BB    ld bc,BBEA        GRA PLOT ABSOLUTE, <de>=x, <hl>=y
C4D3  18 03       jr C4D8

----- command: PLOTR <xd>,<yd>[,<ink>]
      @ DE7B
C4D5  01 ED BB    ld bc,BBED        GRA PLOT RELATIVE, <de>=xd, <hl>=yd
C4D8  C5          push bc
C4D9  CD 1A C5     call C51A        get integer VAL to <de>, next to <bc>
C4DC  CD 55 DD     call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
C4DF  30 06        jr nc,C4E7       no more argument
C4E1  CD 4B C2     call C24B        get VAL into <a>, max=15.; else error
```

```
C4E4  CD DE BB     call BBDE        GRA SET PEN, <a>=ink
C4E7  18 28        jr C511

----- function: TESTR(<xd>,<yd>)
      @ D1A6
C4E9  01 F0 BB     ld bc,BBF0       GRA TEST ABSOLUTE, <de>=x, <hl>=y
C4EC  18 03        jr C4F1

----- function: TEST(<x>,<y>)
      @ D1A8
C4EE  01 F3 BB     ld bc,BBF3       GRA TEST RELATIVE, <de>=xd, <hl>=yd
C4F1  C5           push bc
C4F2  CD 1A C5      call C51A       get integer VAL to <de>, next to <bc>
C4F5  CD 37 DD      call DD37       CHRNEXT <a>, nz=Error; CHRGET
C4F8  29            ')
C4F9  E3            ex (sp),hl
C4FA  C5            push bc
C4FB  E3             ex (sp),hl
C4FC  C1            pop bc
C4FD  CD F9 FF      call FFF9       jp(bc)
C500  CD 0A FF      call FF0A       set FAC to <a> and mark integer
C503  E1           pop hl
C504  C9           ret

----- command: MOVE <x>, <y>
      @ DE5D
C505  01 C0 BB     ld bc,BBC0       GRA MOVE ABSOLUTE, <de>=x, <hl>=y
C508  18 03        jr C50D

----- command: MOVER <xd>,<yd>
      @ DE5F
C50A  01 C3 BB     ld bc,BBC3       GRA MOVE RELATIVE, <de>=xd, <hl>=yd
C50D  C5           push bc
C50E  CD 1A C5      call C51A       get integer VAL to <de>, next to <bc>
C511  E3            ex (sp),hl
C512  C5            push bc
C513  E3             ex (sp),hl
C514  C1            pop bc
C515  CD F9 FF      call FFF9       jp(bc)
C518  E1           pop hl
C519  C9           ret

----- get integer VAL to <de>, next to <bc>
      @ C48C! C496! C49F! C4D9! C4F2! C50E!
C51A  CD 86 CE     call CE86        get integer VAL(expression) in <de>
C51D  D5           push de
C51E  CD 37 DD      call DD37       CHRNEXT <a>, nz=Error; CHRGET
C521  2C            ',
C522  CD 86 CE      call CE86       get integer VAL(expression) in <de>
C525  42            ld b,d
C526  4B            ld c,e
C527  D1           pop de
C528  C9           ret

----- command: FOR <variable> = <start> TO <end> [STEP <step>]
      @ DE3D
C529  CD B3 D6     call D6B3        used by FOR
C52C  E5           push hl
C52D  C5            push bc
C52E  D5             push de
C52F  CD C5 C9        call C9C5     check whether FOR/NEXT match
C532  22 2C AC        ld (AC2C),hl  used by FOR ??
C535  D5              push de
C536  E5               push hl
C537  EB                ex de,hl
```

```
C538  CD 32 C6      call C632     look for a NEXT entry on the Basic stack
C53B  CC AC F5      call z,F5AC   set BASIC STACK pointer to <hl>
C53E  E1            pop hl
C53F  CD 51 DD      call DD51     CHRGOT <a>; end of statement? =carry
C542  11 00 00      ld de,0000
C545  D4 86 D6      call nc,D686  get address of VARIABLE or subscript
C548  44            ld b,h
C549  4D            ld c,l
C54A  E1            pop hl
C54B  E3            ex (sp),hl
C54C  7A            ld a,d
C54D  B3            or e
C54E  C4 B8 FF      call nz,FFB8  test HL=DE? (try hl-de)
C551  C2 F6 C5      jp nz,C5F6    Error: Unexpected next
C554  EB            ex de,hl
C555  CD D2 DD      call DDD2     get BASIC program counter in <hl>
C558  E3            ex (sp),hl
C559  CD CE DD      call DDCE     set BASIC program counter to <hl>
C55C  E1            pop hl
C55D  F1            pop af        a=b
C55E  E3            ex (sp),hl
C55F  D5            push de
C560  C5            push bc
C561  E5            push hl
C562  01 05 16      ld bc,1605    b=22., c=5 (real)
C565  B9            cp c          is it a real variable?
C566  28 0B         jr z,C573     yes, it is
C568  01 02 10      ld bc,1002    b=16., c=2 (integer)
C56B  B9            cp c
C56C  28 05         jr z,C573     is it an integer?
C56E  1E 0D         ld e,0D       Type mismatch
C570  C3 94 CA      jp CA94       perform ERROR <e> routine

----- yes, it is
C573  78            ld a,b        stack len depending on VARTYPE
C574  CD B0 F5      call F5B0     inc BASIC STACK pointer by <a>, (hl)=next lo
C577  73            ld (hl),e
C578  23            inc hl
C579  72            ld (hl),d
C57A  23            inc hl
C57B  E3            ex (sp),hl
C57C  CD 37 DD      call DD37     CHRNEXT <a>, nz=Error; CHRGET
C57F  EF            [=]
C580  CD FB CE      call CEFB     evaluate (expression), CHRGET, cp 01
C583  79            ld a,c
C584  CD D7 FE      call FED7     test <a>=VARTYPE? if not CINT,CREAL
C587  E5            push hl
C588  21 27 AC      ld hl,AC27    FAC used by FOR
C58B  CD 62 FF      call FF62     copy FAC to (hl)
C58E  E1            pop hl
C58F  CD 37 DD      call DD37     CHRNEXT <a>, nz=Error; CHRGET
C592  EC            [TO]
C593  CD FB CE      call CEFB     evaluate (expression), CHRGET, cp 01
C596  E3            ex (sp),hl
C597  79            ld a,c        <c> was 02 or 05, depending on VARTYPE
C598  CD D7 FE      call FED7     test <a>=VARTYPE? if not CINT,CREAL
C59B  CD 62 FF      call FF62     copy FAC to (hl)
C59E  EB            ex de,hl
C59F  E3            ex (sp),hl
C5A0  EB            ex de,hl
C5A1  21 01 00      ld hl,0001    default STEP = 1
C5A4  CD 0D FF      call FF0D     set FAC to <hl> and mark integer
C5A7  EB            ex de,hl
C5A8  7E            ld a,(hl)
C5A9  FE E6         cp E6         [STEP]
```

```
C5AB   20 06          jr nz,C5B3      STEP not given
C5AD   CD 3F DD       call DD3F       CHRGET <a>, skip blank, cp 01
C5B0   CD FB CE       call CEFB       evaluate (expression), CHRGET, cp 01
C5B3   79             ld a,c          change to VARTYPE given at FOR
C5B4   CD D7 FE       call FED7       test <a>=VARTYPE? if not CINT,CREAL
C5B7   E3             ex (sp),hl
C5B8   CD 62 FF       call FF62       copy FAC to (hl)
C5BB   CD A3 FD       call FDA3       get [SGN] <a> (FF,00,01)
C5BE   EB             ex de,hl
C5BF   77             ld (hl),a       store SGN of STEP argument
C5C0   23             inc hl
C5C1   EB             ex de,hl
C5C2   E1            pop hl
C5C3   CD 4A DD      call DD4A        CHRGOT <a>; end of statement? else syntax er
C5C6   EB            ex de,hl
C5C7   73            ld (hl),e
C5C8   23            inc hl
C5C9   72            ld (hl),d
C5CA   23            inc hl
C5CB   EB            ex de,hl
C5CC   CD D2 DD      call DDD2        get BASIC program counter in <hl>
C5CF   EB            ex de,hl
C5D0   73            ld (hl),e
C5D1   23            inc hl
C5D2   72            ld (hl),d
C5D3   23            inc hl
C5D4   D1           pop de
C5D5   73           ld (hl),e
C5D6   23           inc hl
C5D7   72           ld (hl),d
C5D8   23           inc hl
C5D9   ED 5B 2C AC  ld de,(AC2C)     used by FOR ??
C5DD   73           ld (hl),e
C5DE   23           inc hl
C5DF   72           ld (hl),d
C5E0   23           inc hl
C5E1   70           ld (hl),b
C5E2   D1          pop de
C5E3   21 27 AC    ld hl,AC27        FAC used by FOR
C5E6   CD 66 FF    call FF66         copy variable (hl) to (de)
C5E9   AF          xor a             reset flag
C5EA   32 26 AC    ld (AC26),a       flag used by FOR
C5ED   E1         pop hl
C5EE   CD CE DD   call DDCE          set BASIC program counter to <hl>
C5F1   2A 2C AC   ld hl,(AC2C)       used by FOR ??
C5F4   18 0A      jr C600

----- Error: Unexpected next
C5F6   1E 01      ld e,01            Unexpected NEXT
C5F8   C3 94 CA   jp CA94            perform ERROR <e> routine

----- command: NEXT [<list of <variable>>]
       @ DE61
C5FB   3E FF      ld a,FF            set flag
C5FD   32 26 AC   ld (AC26),a        flag used by FOR

       @ C5F4' C62F'
C600   EB         ex de,hl
C601   CD 32 C6   call C632          look for a NEXT entry on the Basic stack
C604   20 F0      jr nz,C5F6         Error: Unexpected next
C606   EB         ex de,hl
C607   CD AC F5   call F5AC          set BASIC STACK pointer to <hl>
C60A   EB         ex de,hl
C60B   E5         push hl
C60C   CD 61 C6    call C661         ...
```

```
C60F  28 0F       jr z,C620        ...
C611  F1          pop af
C612  23          inc hl
C613  5E          ld e,(hl)
C614  23          inc hl
C615  56          ld d,(hl)
C616  23          inc hl
C617  7E          ld a,(hl)
C618  23          inc hl
C619  66          ld h,(hl)
C61A  6F          ld l,a
C61B  CD CE DD    call DDCE        set BASIC program counter to <hl>
C61E  EB          ex de,hl
C61F  C9          ret

C620  01 05 00     ld bc,0005      ...
C623  09           add hl,bc
C624  5E           ld e,(hl)
C625  23           inc hl
C626  56           ld d,(hl)
C627  E1          pop hl
C628  CD AC F5    call F5AC        set BASIC STACK pointer to <hl>
C62B  EB          ex de,hl
C62C  CD 55 DD    call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
C62F  38 CF       jr c,C600        do the next variable
C631  C9          ret

----- look for a NEXT entry on the Basic stack
      @ C538! C601!
C632  2A 8B B0    ld hl,(B08B)     BASIC STACK pointer
C635  E5          push hl
C636  2B           dec hl
C637  46           ld b,(hl)
C638  23           inc hl
C639  7D           ld a,l
C63A  90           sub b
C63B  6F           ld l,a
C63C  9F           sbc a,a
C63D  84           add a,h
C63E  67           ld h,a
C63F  E3           ex (sp),hl
C640  78           ld a,b
C641  FE 07        cp 07           =7.
C643  28 19        jr z,C65E       ...
C645  FE 10        cp 10           =16.
C647  28 04        jr z,C64D       ...
C649  FE 16        cp 16           =22.
C64B  20 0D        jr nz,C65A      ...
C64D  E5           push hl
C64E  2B            dec hl
C64F  2B            dec hl
C650  7E            ld a,(hl)
C651  2B            dec hl
C652  6E            ld l,(hl)
C653  67            ld h,a
C654  CD B8 FF      call FFB8      test HL=DE? (try hl-de)
C657  E1           pop hl
C658  20 04        jr nz,C65E      ...
C65A  EB           ex de,hl
C65B  E1          pop hl
C65C  78          ld a,b
C65D  C9          ret
```

```
C65E  E1         pop hl
C65F  18 D4      jr C635          ...

      @ C60C!
C661  5E         ld e,(hl)
C662  23         inc hl
C663  56         ld d,(hl)
C664  23         inc hl
C665  FE 10      cp 10            =16.
C667  28 2D      jr z,C696        ...
C669  E5         push hl
C66A  01 05 00    ld bc,0005      ...
C66D  79          ld a,c
C66E  EB          ex de,hl
C66F  CD 4B FF    call FF4B       set VARTYPE <a>, copy VARIABLE (hl) to FAC
C672  E1         pop hl
C673  3A 26 AC   ld a,(AC26)      flag used by FOR
C676  B7         or a
C677  28 10      jr z,C689        ...
C679  E5         push hl
C67A  09          add hl,bc
C67B  CD CC FC    call FCCC       perform [+] (plus)
C67E  E1         pop hl
C67F  E5         push hl
C680  2B          dec hl
C681  56          ld d,(hl)
C682  2B          dec hl
C683  5E          ld e,(hl)
C684  EB          ex de,hl
C685  CD 62 FF    call FF62       copy FAC to (hl)
C688  E1         pop hl
C689  E5         push hl
C68A  0E 05       ld c,05         =5.
C68C  CD 09 FD    call FD09       compare two numbers (int or real)
C68F  E1         pop hl
C690  01 0A 00   ld bc,000A       =10.
C693  09         add hl,bc
C694  96         sub (hl)
C695  C9         ret

C696  E5         push hl
C697  EB          ex de,hl
C698  5E          ld e,(hl)
C699  23          inc hl
C69A  56          ld d,(hl)
C69B  3A 26 AC    ld a,(AC26)     flag used by FOR
C69E  B7          or a
C69F  28 16       jr z,C6B7       ...
C6A1  E3          ex (sp),hl
C6A2  E5          push hl
C6A3  23           inc hl
C6A4  23           inc hl
C6A5  7E           ld a,(hl)
C6A6  23           inc hl
C6A7  66           ld h,(hl)
C6A8  6F           ld l,a
C6A9  CD AC BD     call BDAC      INT ARITH ADD; <hl>=<hl>+<de>
C6AC  1E 06        ld e,06        Overflow
C6AE  D2 94 CA     jp nc,CA94     perform ERROR <e> routine
C6B1  EB           ex de,hl
C6B2  E1          pop hl
C6B3  E3          ex (sp),hl
C6B4  72          ld (hl),d
C6B5  2B          dec hl
C6B6  73          ld (hl),e
```

```
C6B7  E1          pop hl
C6B8  7E          ld a,(hl)
C6B9  23          inc hl
C6BA  E5          push hl
C6BB  66           ld h,(hl)
C6BC  6F           ld l,a
C6BD  EB           ex de,hl
C6BE  CD C4 BD     call BDC4       INT ARITH, COMPARE <hl>,<de>; <a>= FF,00,01
C6C1  E1          pop hl
C6C2  23          inc hl
C6C3  23          inc hl
C6C4  23          inc hl
C6C5  96          sub (hl)
C6C6  C9          ret

----- command: IF <logic expr>
      @ DE43
C6C7  CD FB CE    call CEFB       evaluate (expression), CHRGET, cp 01
C6CA  FE A0       cp A0           [GOTO]
C6CC  28 04       jr z,C6D2
C6CE  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
C6D1  EB          [THEN]
C6D2  E5          push hl
C6D3  CD A3 FD     call FDA3      get [SGN] <a> (FF,00,01)
C6D6  E1          pop hl
C6D7  CC 9F E8    call z,E89F     skip over statements till [ELSE] or <line en
C6DA  C8          ret z
C6DB  CD 51 DD    call DD51       CHRGOT <a>; end of statement? =carry
C6DE  D8          ret c
C6DF  FE 1E       cp 1E           <next LINE#>
C6E1  28 05       jr z,C6E8       command: GOTO <line#>
C6E3  FE 1D       cp 1D           <next ADDRESS>
C6E5  C2 AB DD    jp nz,DDAB      look for other tokens

----- command: GOTO <line#>
      @ C6E1' DE41
C6E8  CD 67 E7    call E767       get address VAL into <de>
C6EB  EB          ex de,hl
C6EC  C9          ret

----- command: GOSUB <line#>
      @ DE3F
C6ED  CD 67 E7    call E767       get address VAL into <de>
C6F0  CD EF E8    call E8EF       command: DATA <list of<data>>   (skip this 1
C6F3  EB          ex de,hl
C6F4  0E 00       ld c,00         flag for GOSUB

----- save PC and register <c>,<de> on BASIC STACK
      @ C887!
C6F6  E5          push hl
C6F7  3E 06        ld a,06        uses 6 bytes on BASIC STACK
C6F9  CD B0 F5     call F5B0      inc BASIC STACK pointer by <a>, (hl)=next lo
C6FC  71           ld (hl),c      marks BASIC STACK for GOSUB
C6FD  23           inc hl
C6FE  73           ld (hl),e
C6FF  23           inc hl
C700  72           ld (hl),d      save <de> on stack
C701  23           inc hl
C702  EB           ex de,hl
C703  CD D2 DD     call DDD2      get BASIC program counter in <hl>
C706  EB           ex de,hl
C707  73           ld (hl),e
C708  23           inc hl
C709  72           ld (hl),d      save return addr on BASIC STACK
C70A  23           inc hl
```

```
C70B  36 06         ld (hl),06       marks len used
C70D  E1           pop hl
C70E  C9           ret

----- command: RETURN
      @ DE93
C70F  C0           ret nz
C710  CD 2E C7     call C72E        find RETURN on BASIC STACK
C713  CD AC F5     call F5AC        set BASIC STACK pointer to <hl>
C716  4E           ld c,(hl)
C717  23           inc hl
C718  5E           ld e,(hl)
C719  23           inc hl
C71A  56           ld d,(hl)
C71B  23           inc hl
C71C  7E           ld a,(hl)
C71D  23           inc hl
C71E  66           ld h,(hl)
C71F  6F           ld l,a
C720  CD CE DD     call DDCE        set BASIC program counter to <hl>
C723  EB           ex de,hl
C724  79           ld a,c
C725  FE 01        cp 01            flag for GOSUB?
C727  D8           ret c            yes, return
C728  CA A4 C8     jp z,C8A4        perform RETURN from EVERY/AFTER
C72B  C3 B6 C8     jp C8B6          perform normal RETURN, part 2

----- find RETURN on BASIC STACK
      @ C710!
C72E  2A 8B B0     ld hl,(B08B)     BASIC STACK pointer
C731  2B           dec hl
C732  7E           ld a,(hl)        len of entry
C733  F5           push af
C734  7D            ld a,l
C735  96            sub (hl)
C736  6F            ld l,a
C737  9F            sbc a,a
C738  84            add a,h
C739  67            ld h,a
C73A  23            inc hl
C73B  F1           pop af
C73C  FE 06        cp 06            was len =6?
C73E  C8           ret z            yes, return
C73F  B7           or a
C740  20 EF        jr nz,C731       no, step up to next entry
C742  1E 03        ld e,03          Unexpected RETURN
C744  C3 94 CA     jp CA94          perform ERROR <e> routine

----- command: WHILE <logic expression>
      @ DEAD
C747  E5           push hl
C748  CD 18 CA      call CA18       called from WHILE
C74B  E5            push hl
C74C  EB             ex de,hl
C74D  22 2E AC       ld (AC2E),hl   used by WHILE, WEND
C750  CD B8 C7       call C7B8      find a WHILE entry on the Basic stack
C753  CC AC F5       call z,F5AC    set BASIC STACK pointer to <hl>
C756  3E 07          ld a,07        len of entry for WHILE
C758  CD B0 F5       call F5B0      inc BASIC STACK pointer by <a>, (hl)=next lo
C75B  EB             ex de,hl
C75C  CD D2 DD       call DDD2      get BASIC program counter in <hl>
C75F  EB             ex de,hl
C760  73             ld (hl),e
C761  23             inc hl
C762  72             ld (hl),d
```

```
C763  23          inc hl
C764  D1          pop de
C765  73          ld (hl),e
C766  23          inc hl
C767  72          ld (hl),d
C768  23          inc hl
C769  EB          ex de,hl
C76A  E3          ex (sp),hl
C76B  EB          ex de,hl
C76C  73          ld (hl),e
C76D  23          inc hl
C76E  72          ld (hl),d
C76F  23          inc hl
C770  36 07       ld (hl),07
C772  EB          ex de,hl
C773  D1          pop de
C774  18 2A       jr C7A0         ...

----- command: WEND
      @ DEAB
C776  C0          ret nz
C777  EB          ex de,hl
C778  CD B8 C7    call C7B8       find a WHILE entry on the Basic stack
C77B  1E 1E       ld e,1E         Unexpected WEND
C77D  C2 94 CA    jp nz,CA94      perform ERROR <e> routine
C780  E5          push hl
C781  11 07 00     ld de,0007     len of WHILE entry on Basic stack
C784  19           add hl,de
C785  CD AC F5     call F5AC      set BASIC STACK pointer to <hl>
C788  CD D2 DD     call DDD2      get BASIC program counter in <hl>
C78B  22 2E AC     ld (AC2E),hl   used by WHILE, WEND
C78E  E1          pop hl
C78F  5E          ld e,(hl)
C790  23          inc hl
C791  56          ld d,(hl)
C792  23          inc hl
C793  EB          ex de,hl
C794  CD CE DD    call DDCE       set BASIC program counter to <hl>
C797  EB          ex de,hl
C798  5E          ld e,(hl)
C799  23          inc hl
C79A  56          ld d,(hl)
C79B  23          inc hl
C79C  7E          ld a,(hl)
C79D  23          inc hl
C79E  66          ld h,(hl)
C79F  6F          ld l,a
C7A0  D5          push de
C7A1  CD FB CE     call CEFB      evaluate (expression), CHRGET, cp 01
C7A4  E5           push hl
C7A5  CD A3 FD      call FDA3     get [SGN] <a> (FF,00,01)
C7A8  E1           pop hl
C7A9  D1          pop de
C7AA  C0          ret nz
C7AB  2A 2E AC    ld hl,(AC2E)    used by WHILE, WEND
C7AE  CD CE DD    call DDCE       set BASIC program counter to <hl>
C7B1  3E 07       ld a,07         len of entry for WHILE
C7B3  CD A0 F5    call F5A0       decrement BASIC STACK pointer by <a>
C7B6  EB          ex de,hl
C7B7  C9          ret
```

```
----- find a WHILE entry on the Basic stack
      @ C750! C778!
C7B8  2A 8B B0     ld hl,(B08B)      BASIC STACK pointer
C7BB  2B           dec hl
C7BC  E5           push hl
C7BD  7D            ld a,l
C7BE  96            sub (hl)
C7BF  6F            ld l,a
C7C0  9F            sbc a,a
C7C1  84            add a,h
C7C2  67            ld h,a
C7C3  23            inc hl
C7C4  E3            ex (sp),hl
C7C5  7E            ld a,(hl)
C7C6  FE 10         cp 10            =16.
C7C8  28 16         jr z,C7E0        it's a FOR integer entry
C7CA  FE 16         cp 16            =22.
C7CC  28 12         jr z,C7E0        it's a FOR real entry
C7CE  FE 07         cp 07            is it a WHILE entry?
C7D0  20 0C         jr nz,C7DE       no, it's not
C7D2  2B            dec hl
C7D3  2B            dec hl
C7D4  2B            dec hl
C7D5  7E            ld a,(hl)
C7D6  2B            dec hl
C7D7  6E            ld l,(hl)
C7D8  67            ld h,a
C7D9  CD B8 FF      call FFB8        test HL=DE? (try hl-de)
C7DC  20 02         jr nz,C7E0       ...
C7DE  E1           pop hl
C7DF  C9           ret

C7E0  E1           pop hl
C7E1  18 D8        jr C7BB           ...

----- command: ON
      @ DE65
C7E3  FE 9C        cp 9C             [ERROR]
C7E5  CA E5 CB     jp z,CBE5         here: ON ERROR

----- here: ON <expression>
C7E8  CD 67 CE     call CE67         get byte VAL(expression) in <de>
C7EB  4F           ld c,a            = result
C7EC  46           ld b,(hl)         get token
C7ED  78           ld a,b
C7EE  FE A0        cp A0             [GOTO]
C7F0  28 05        jr z,C7F7         here: ON <expression> GOTO or GOSUB
C7F2  CD 37 DD     call DD37         CHRNEXT <a>, nz=Error; CHRGET
C7F5  9F           [GOSUB]
C7F6  2B           dec hl

----- here: ON <expression> GOTO or GOSUB
      @ C7F0' C804'
C7F7  0D           dec c
C7F8  78           ld a,b            =token
C7F9  CA AB DD     jp z,DDAB         look for other tokens
C7FC  CD 3F DD     call DD3F         CHRGET <a>, skip blank, cp 01
C7FF  CD E1 CE     call CEE1         get line# into <de>
C802  FE 2C        cp 2C             ',
C804  28 F1        jr z,C7F7         here: ON <expression> GOTO or GOSUB
C806  C9           ret
```

```
----- there is a higher priority
      @ DD82!
C807  AF          xor a              reset flag
C808  32 30 AC    ld (AC30),a        used by ON
C80B  CD FB BC    call BCFB          KL NEXT SYNC, =(hl), =<a> prev. prio, =carry
C80E  30 1D       jr nc,C82D         no more event
C810  47          ld b,a
C811  3A 30 AC    ld a,(AC30)        used by ON
C814  E6 7F       and 7F             clear bit 7
C816  32 30 AC    ld (AC30),a        used by ON
C819  C5          push bc
C81A  E5           push hl
C81B  CD FE BC      call BCFE        KL DO SYNC, perform SYNC EVENT block (hl)
C81E  E1           pop hl
C81F  C1          pop bc
C820  3A 30 AC    ld a,(AC30)        used by ON
C823  17          rla
C824  F5          push af
C825  78           ld a,b
C826  D4 01 BD     call nc,BD01      KL DONE SYNC, (hl)=block, <a>=prev. priority
C829  F1          pop af
C82A  17          rla
C82B  30 DE       jr nc,C80B         next
C82D  3A 30 AC    ld a,(AC30)        used by ON
C830  E6 04       and 04             mask out
C832  C4 53 C4    call nz,C453       establish BREAK EVENT
C835  2A 34 AE    ld hl,(AE34)       program counter on RUN
C838  3A 30 AC    ld a,(AC30)        used by ON
C83B  E6 03       and 03             mask out
C83D  C8          ret z
C83E  1F          rra
C83F  DA 6B CB    jp c,CB6B          perform a BREAK
C842  23          inc hl
C843  F1          pop af
C844  C3 93 DD    jp DD93            RUN  LOOP, part 2

----- event routine BREAK, part 2
      @ C46C
C847  22 36 AC    ld (AC36),hl       save for Basic PC on BREAK (within event blo
C84A  3E 04       ld a,04            =4.
C84C  30 50       jr nc,C89E         ...
C84E  2A 34 AC    ld hl,(AC34)       line# for ON BREAK GOSUB
C851  7C          ld a,h
C852  B5          or l
C853  C4 D6 DD    call nz,DDD6       get BASIC line# at PC in <hl>, =carry
C856  3E 41       ld a,41            =65.
C858  30 44       jr nc,C89E         ...
C85A  11 31 AC    ld de,AC31         ...
C85D  0E 02       ld c,02            =2.
C85F  18 25       jr C886            ...

----- save subroutine addr @ queue+10.
      @ C952! C993!
C861  D5          push de
C862  CD 37 DD     call DD37         CHRNEXT <a>, nz=Error; CHRGET
C865  9F           [GOSUB]
C866  CD 67 E7     call E767         get address VAL into <de>
C869  42           ld b,d
C86A  4B           ld c,e            bc=de
C86B  CD 61 DD     call DD61         CHRSKIP <a>; skip over blank, tab, linefeed
C86E  D1          pop de
C86F  E5          push hl
C870  21 0A 00     ld hl,000A        =10.
C873  19           add hl,de
C874  71           ld (hl),c
```

```
C875  23            inc hl
C876  70            ld (hl),b       store SUBROUTINE address
C877  E1            pop hl
C878  C9            ret

----- event routine TIMER
      @ AC66: AC78: AC80: AC92: C928:
C879  23            inc hl
C87A  23            inc hl
C87B  23            inc hl
C87C  EB            ex de,hl
C87D  CD D6 DD      call DDD6       get BASIC line# at PC in <hl>, =carry
C880  3E 40         ld a,40         =64.
C882  30 1A         jr nc,C89E      ...
C884  0E 01         ld c,01         =1.
C886  D5            push de
C887  CD F6 C6      call C6F6       save PC and register <c>,<de> on BASIC STACK
C88A  2A 34 AE      ld hl,(AE34)    program counter on RUN
C88D  EB            ex de,hl
C88E  E1            pop hl
C88F  70            ld (hl),b
C890  23            inc hl
C891  73            ld (hl),e
C892  23            inc hl
C893  72            ld (hl),d
C894  23            inc hl
C895  5E            ld e,(hl)
C896  23            inc hl
C897  56            ld d,(hl)
C898  EB            ex de,hl
C899  22 34 AE      ld (AE34),hl    program counter on RUN
C89C  3E C2         ld a,C2         =194.

      @ C84C' C858' C882'
C89E  21 30 AC      ld hl,AC30      used by ON
C8A1  B6            or (hl)
C8A2  77            ld (hl),a
C8A3  C9            ret

----- perform RETURN from EVERY/AFTER
      @ C728
C8A4  7E            ld a,(hl)
C8A5  23            inc hl
C8A6  5E            ld e,(hl)
C8A7  23            inc hl
C8A8  56            ld d,(hl)
C8A9  D5            push de
C8AA  01 F7 FF      ld bc,FFF7      = -9
C8AD  09            add hl,bc
C8AE  CD 01 BD      call BD01       KL DONE SYNC, (hl)=block, <a>=prev. priority
C8B1  E1            pop hl
C8B2  F1            pop af
C8B3  C3 74 DD      jp DD74         do the RUN LOOP

----- perform normal RETURN, part 2
      @ C72B
C8B6  7E            ld a,(hl)
C8B7  2A 36 AC      ld hl,(AC36)    save for Basic PC on BREAK (within event blo
C8BA  01 FC FF      ld bc,FFFC      = -4
C8BD  09            add hl,bc
C8BE  CD 01 BD      call BD01       KL DONE SYNC, (hl)=block, <a>=prev. priority
C8C1  CD 53 C4      call C453       establish BREAK EVENT
C8C4  2A 32 AC      ld hl,(AC32)    ...
C8C7  F1            pop af
C8C8  C3 74 DD      jp DD74         do the RUN LOOP
```

```
----- command: ON BREAK
      @ DE67
C8CB  FE CE        cp CE              [STOP]
C8CD  11 00 00     ld de,0000         reset line# to 0000
C8D0  28 08        jr z,C8DA          ON BREAK STOP
C8D2  CD 37 DD     call DD37          CHRNEXT <a>, nz=Error; CHRGET
C8D5  9F           [GOSUB]
C8D6  CD 67 E7     call E767          get address VAL into <de>
C8D9  2B           dec hl
C8DA  ED 53 34 AC  ld (AC34),de       line# for ON BREAK GOSUB
C8DE  C3 3F DD     jp DD3F            CHRGET <a>, skip blank, cp 01

----- command: DI
      @ DEB7
C8E1  E5           push hl
C8E2  CD 04 BD      call BD04         KL EVENT DISABLE
C8E5  E1           pop hl
C8E6  C9           ret

----- command: EI
      @ DEB9
C8E7  E5           push hl
C8E8  CD 07 BD      call BD07         KL EVENT ENABLE
C8EB  E1           pop hl
C8EC  C9           ret

----- set up all default events
      @ C180!
C8ED  CD A7 BC     call BCA7          SOUND RESET
C8F0  21 5C AC     ld hl,AC5C         TIMER, block #0  (4 blocks total)
C8F3  06 04        ld b,04            count of blocks to remove
C8F5  E5           push hl
C8F6  CD EC BC      call BCEC         KL DEL TICKER, remove block (hl) from tick 1
C8F9  E1            pop hl
C8FA  11 12 00      ld de,0012        displacement to next entry
C8FD  19            add hl,de
C8FE  10 F5         djnz C8F5         delete next ticker
C900  CD 48 BB     call BB48          KM DISARM BREAK
C903  CD F5 BC     call BCF5          KL SYNC RESET, clear synchronous event queue
C906  21 00 00     ld hl,0000         reset line#
C909  22 34 AC     ld (AC34),hl       line# for ON BREAK GOSUB
C90C  CD 53 C4     call C453          establish BREAK EVENT
C90F  21 38 AC     ld hl,AC38         sound chan 1 (bit 0)
C912  11 05 03     ld de,0305         <d>=3 sound events (3 sound channels)
C915  01 00 08     ld bc,0800         <b>=class for SOUND event
C918  CD 24 C9     call C924          initialise all event blocks
C91B  21 62 AC     ld hl,AC62         timer #0 +6= event block timer #0
C91E  11 0B 04     ld de,040B         <d>=4 timers
C921  01 01 02     ld bc,0201         <b>=class for TIMER event #0

----- initialise all event blocks
      @ C918! C93D'
C924  C5           push bc
C925  D5            push de
C926  0E FD          ld c,FD          enable upper, disable lower ROM
C928  11 79 C8       ld de,C879       event routine TIMER
C92B  CD EF BC       call BCEF        KL INIT EVENT BLOCK (hl)=block, <b>=class, <
C92E  D1            pop de
C92F  D5            push de
C930  16 00          ld d,00
C932  19             add hl,de        <e>=displacement to next block
C933  D1            pop de
C934  C1           pop bc
C935  79           ld a,c
C936  B7           or a
```

```
C937  28 03       jr z,C93C        do not change priority for sound chan
C939  78          ld a,b
C93A  87          add a,a          <b>=2*<b>; increase timer priority
C93B  47          ld b,a
C93C  15          dec d            <d>=count of blocks
C93D  20 E5       jr nz,C924       next block
C93F  C9          ret

----- command: ON SQ(<sound channnel>) GOSUB
      @ DE6B
C940  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
C943  28          '(
C944  CD 67 CE    call CE67        get byte VAL(expression) in <de>
C947  F5          push af
C948  CD 5D C9     call C95D       check for legal SOUND chan
C94B  B7           or a
C94C  20 1E        jr nz,C96C      Error: Improper argument
C94E  CD 37 DD     call DD37       CHRNEXT <a>, nz=Error; CHRGET
C951  29           ')
C952  CD 61 C8     call C861       save subroutine addr @ queue+10.
C955  F1          pop af
C956  E5          push hl
C957  EB           ex de,hl
C958  CD B0 BC     call BCB0       SOUND ARM EVENT, <a>=channels, (hl)=event bl
C95B  E1          pop hl
C95C  C9          ret

----- check for legal SOUND chan
      @ C948!
C95D  1F          rra
C95E  11 38 AC    ld de,AC38       sound chan 1 (bit 0)
C961  D8          ret c
C962  1F          rra
C963  11 44 AC    ld de,AC44       sound chan 2 (bit 1)
C966  D8          ret c
C967  1F          rra
C968  11 50 AC    ld de,AC50       sound chan 3 (bit 2)
C96B  D8          ret c

----- Error: Improper argument
C96C  1E 05        ld e,05         Improper argument
C96E  C3 94 CA     jp CA94         perform ERROR <e> routine

----- command: AFTER <time period> [,<timer>] GOSUB <line#>
      @ DE01
C971  CD 7C CE    call CE7C        get integer VAL of expression, neg=error
C974  01 00 00    ld bc,0000
C977  18 05       jr C97E

----- command: EVERY <time period> [,<timer>] GOSUB <line#>
      @ DE3B
C979  CD 7C CE    call CE7C        get integer VAL of expression, neg=error
C97C  42          ld b,d
C97D  4B          ld c,e           bc=de
C97E  D5          push de
C97F  C5           push bc
C980  CD 55 DD      call DD55      CHRBACK comma?; if=:CHRGET <a>, scf
C983  11 00 00      ld de,0000     default timer 0
C986  DC 86 CE      call c,CE86    get integer VAL(expression) in <de>
C989  EB            ex de,hl
C98A  CD B1 C9      call C9B1      get address of TIMER BLOCK
C98D  E5            push hl
C98E  01 06 00       ld bc,0006    len of TICKER data
C991  09             add hl,bc
C992  EB             ex de,hl
```

```
C993  CD 61 C8      call C861      save subroutine addr @ queue+10.
C996  D1            pop de
C997  C1            pop bc
C998  E3            ex (sp),hl
C999  EB            ex de,hl
C99A  CD E9 BC      call BCE9      KL ADD TICKER, (hl)=tick block, <de>=initial
C99D  E1            pop hl
C99E  C9            ret

----- function: REMAIN(<timer>)
      @ D1D4
C99F  CD 8D FE      call FE8D      function: CINT(<num expression>) in <hl>
C9A2  CD B1 C9      call C9B1      get address of TIMER BLOCK
C9A5  CD EC BC      call BCEC      KL DEL TICKER, remove block (hl) from tick l
C9A8  38 03         jr c,C9AD      result valid
C9AA  11 00 00      ld de,0000     timer not initialised
C9AD  EB            ex de,hl
C9AE  C3 0D FF      jp FF0D        set FAC to <hl> and mark integer

----- get address of TIMER BLOCK
      @ C98A! C9A2!
C9B1  7C            ld a,h
C9B2  B7            or a           test <h> for zero
C9B3  20 B7         jr nz,C96C     Error: Improper argument
C9B5  7D            ld a,l
C9B6  FE 04         cp 04          max VAL =3
C9B8  30 B2         jr nc,C96C     Error: Improper argument
C9BA  87            add a,a
C9BB  87            add a,a
C9BC  87            add a,a
C9BD  85            add a,l
C9BE  87            add a,a        <l>=<l>*18
C9BF  6F            ld l,a
C9C0  01 5C AC      ld bc,AC5C     TIMER, block #0  (4 blocks total)
C9C3  09            add hl,bc      add displacement
C9C4  C9            ret

----- check whether FOR/NEXT match
      @ C52F!
C9C5  EB            ex de,hl
C9C6  CD D2 DD      call DDD2      get BASIC program counter in <hl>
C9C9  EB            ex de,hl
C9CA  2B            dec hl
C9CB  06 01         ld b,01
C9CD  0E 1A         ld c,1A        NEXT missing
C9CF  CD 23 E9      call E923      test line for ELSE or THEN; error if pgm end
C9D2  E5            push hl
C9D3  CD 3F DD      call DD3F      CHRGET <a>, skip blank, cp 01
C9D6  FE B0         cp B0          [NEXT]
C9D8  28 08         jr z,C9E2      found
C9DA  E1            pop hl
C9DB  FE 9E         cp 9E          [FOR]
C9DD  20 EE         jr nz,C9CD     no, it's not [FOR]
C9DF  04            inc b          inc count for [FOR]
C9E0  18 EB         jr C9CD        continue check

C9E2  F1            pop af
C9E3  EB            ex de,hl
C9E4  E5            push hl
C9E5  CD D2 DD      call DDD2      get BASIC program counter in <hl>
C9E8  E3            ex (sp),hl
C9E9  CD CE DD      call DDCE      set BASIC program counter to <hl>
C9EC  EB            ex de,hl
C9ED  05            dec b
C9EE  28 24         jr z,CA14      FOR/NEXT match, return
```

```
C9F0  CD 3F DD     call DD3F       CHRGET <a>, skip blank, cp 01
C9F3  28 0E        jr z,CA03       end of statement
C9F5  C5           push bc
C9F6  D5            push de
C9F7  CD 86 D6       call D686     get address of VARIABLE or subscript
C9FA  D1            pop de
C9FB  C1           pop bc
C9FC  CD 55 DD     call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
C9FF  30 02        jr nc,CA03      not a comma
CA01  10 F2        djnz C9F5       get next variable
CA03  2B           dec hl
CA04  78           ld a,b
CA05  B7           or a
CA06  28 0C        jr z,CA14       FOR/NEXT match, return
CA08  EB           ex de,hl
CA09  CD D2 DD     call DDD2       get BASIC program counter in <hl>
CA0C  E3           ex (sp),hl
CA0D  CD CE DD     call DDCE       set BASIC program counter to <hl>
CA10  E1          pop hl
CA11  EB          ex de,hl
CA12  18 B9       jr C9CD          check next line

CA14  D1          pop de
CA15  C3 3F DD    jp DD3F          CHRGET <a>, skip blank, cp 01

----- called from WHILE
      @ C748!
CA18  2B          dec hl
CA19  EB          ex de,hl
CA1A  CD D2 DD    call DDD2        get BASIC program counter in <hl>
CA1D  EB          ex de,hl
CA1E  06 00       ld b,00          init counter for WHILE and WEND
CA20  04          inc b
CA21  0E 1D       ld c,1D          WEND missing
CA23  CD 23 E9    call E923        test line for ELSE or THEN; error if pgm end
CA26  E5          push hl
CA27  CD 3F DD     call DD3F       CHRGET <a>, skip blank, cp 01
CA2A  E1          pop hl
CA2B  FE D6       cp D6            [WHILE]
CA2D  28 F1       jr z,CA20        WHILE found, increment count
CA2F  FE D5       cp D5            [WEND]
CA31  20 EE       jr nz,CA21       not a WEND, try next
CA33  10 EC       djnz CA21        WEND found, decrement count
CA35  CD 3F DD    call DD3F        CHRGET <a>, skip blank, cp 01
CA38  C3 3F DD    jp DD3F          CHRGET <a>, skip blank, cp 01

----- put 0 in edit buffer and read a line
      @ C0B0! C119! D568! DBAD!
CA3B  21 A4 AC    ld hl,ACA4       EDIT BUFFER
CA3E  36 00       ld (hl),00       =0.
CA40  C3 3A BD    jp BD3A          EDI LINE EDITOR (hl)

----- keyboard edit line in edit buffer
      @ C05F!
CA43  21 A4 AC    ld hl,ACA4       EDIT BUFFER
CA46  CD 3A BD    call BD3A        EDI LINE EDITOR (hl)
CA49  C3 4E C3    jp C34E          output 'LF to channel

----- read a line from tape to edit buffer
      @ EBF5!
CA4C  C5          push bc
CA4D  D5           push de
CA4E  21 A4 AC      ld hl,ACA4     EDIT BUFFER
CA51  E5            push hl
CA52  06 01          ld b,01       char count for input
```

```
CA54  0E 00          ld c,00          =0.
CA56  CD 80 BC       call BC80        CAS IN CHAR from input file
CA59  CA 6B CB       jp z,CB6B        perform a BREAK
CA5C  30 22          jr nc,CA80       nothing on
CA5E  77             ld (hl),a        store char in buffer
CA5F  FE 0D          cp 0D            'CR  (^M)
CA61  28 17          jr z,CA7A        it is a 'CR, replace by 0
CA63  0E 00          ld c,00
CA65  FE 0A          cp 0A            'LF  (^J)
CA67  20 06          jr nz,CA6F       not a linefeed
CA69  78             ld a,b           get count
CA6A  3D             dec a
CA6B  28 E7          jr z,CA54        count =1 before, ignore line feed
CA6D  0E FF          ld c,FF
CA6F  78             ld a,b           get count
CA70  B7             or a
CA71  1E 17          ld e,17          Line too long
CA73  CA 94 CA       jp z,CA94        perform ERROR <e> routine
CA76  23             inc hl           point to next buffer location
CA77  04             inc b            inc count
CA78  18 DC          jr CA56          next char

CA7A  79             ld a,c
CA7B  B7             or a
CA7C  20 D8          jr nz,CA56       next char
CA7E  77             ld (hl),a        store in buffer
CA7F  37             scf
CA80  E1             pop hl
CA81  D1             pop de
CA82  C1             pop bc
CA83  C9             ret

----- reset last error# to 0
      @ C02B!
CA84  AF             xor a            =0

----- store error# and error address
      @ CA98!
CA85  32 AA AD       ld (ADAA),a      last Basic ERROR number
CA88  CD D2 DD       call DDD2        get BASIC program counter in <hl>
CA8B  22 A6 AD       ld (ADA6),hl     ERROR ADDRESS (addr where error occurred)
CA8E  C9             ret

----- command: ERROR <error#>
      @ DE39
CA8F  CD 6D CE       call CE6D        <a>=next VAL, 0=error
CA92  C0             ret nz
CA93  5F             ld e,a

----- perform ERROR <e> routine
      @ C207  C570  C5F8  C6AE  C744  C77D  C96E  CA73  CB02  CB84  CBC8
      @ CC31  CEAD  CFEF  D07D  D122  D13E  D267  D282  D34B  D4A8  D516
      @ D64C  D9DD  DBD0  DC44  DD08  DD28  DDC8  DF2D  E310  E757  E7A0
      @ E8EC  E92D  EB45  EBB5  EC06  EC3A  F1B7  F348  F34D  F5E3  F72B
      @ F740  F86B  F89E  FBCD  FD55  FED4  FF42
CA94  CD 04 AC       call AC04        Indirection: ERROR MESSAGE
CA97  7B             ld a,e
CA98  CD 85 CA       call CA85        store error# and error address
CA9B  2A 34 AE       ld hl,(AE34)     program counter on RUN
CA9E  22 A8 AD       ld (ADA8),hl     program counter on error break
CAA1  CD B0 CB       call CBB0        save pointers if program isn't ended
```

```
----- perform ERROR <e> routine, part 2
      @ CC00
CAA4  31 00 C0     ld sp,C000        init stack pointer
CAA7  2A 32 AE     ld hl,(AE32)      temp storage BASIC STACK pointer
CAAA  CD AC F5     call F5AC         set BASIC STACK pointer to <hl>
CAAD  CD B3 FB     call FBB3         reset string stack
CAB0  CD FD D9     call D9FD         reset FN pointers as not used
CAB3  CD DF CA     call CADF         get next line after error <hl>, if none <hl>
CAB6  2A AF AD     ld hl,(ADAF)      ON ERROR address
CAB9  EB           ex de,hl
CABA  21 B1 AD     ld hl,ADB1        flag ON ERROR
CABD  30 0C        jr nc,CACB        this is an error
CABF  7A           ld a,d
CAC0  B3           or e
CAC1  28 08        jr z,CACB         this is an error
CAC3  A6           and (hl)
CAC4  20 05        jr nz,CACB        this is an error
CAC6  35           dec (hl)
CAC7  EB           ex de,hl          get ON ERROR address
CAC8  C3 93 DD     jp DD93           RUN  LOOP, part 2

----- this is an error
CACB  36 00        ld (hl),00        reset ON ERROR flag
CACD  3A AA AD     ld a,(ADAA)       last Basic ERROR number
CAD0  CD 45 CC     call CC45         get error message text (hl)
CAD3  2A A6 AD     ld hl,(ADA6)      ERROR ADDRESS (addr where error occurred)
CAD6  CD CE DD     call DDCE         set BASIC program counter to <hl>
CAD9  CD 36 CB     call CB36         print text (de); if RUN: 'in <line#>
CADC  C3 64 C0     jp C064           reset Basic

----- get next line after error <hl>, if none <hl>=0000
      @ C08A! CAB3! D0EF!
CADF  2A A6 AD     ld hl,(ADA6)      ERROR ADDRESS (addr where error occurred)
CAE2  CD D9 DD     call DDD9         is there a line#?
CAE5  D8           ret c
CAE6  21 00 00     ld hl,0000        0000
CAE9  C9           ret               when there is no further line

----- Error: Division by zero
      @ D50E  FD31
CAEA  D5           push de
CAEB  E5            push hl
CAEC  21 13 CD       ld hl,CD13      Error message: Division by zero
CAEF  1E 0B          ld e,0B         Division by zero
CAF1  18 07          jr CAFA

----- Error: Overflow
      @ D511  ECD9  ED41  FCDE  FD34
CAF3  D5           push de
CAF4  E5            push hl
CAF5  21 B9 CC       ld hl,CCB9      Error message: Overflow
CAF8  1E 06          ld e,06         Overflow
CAFA  F5             push af
CAFB  E5              push hl
CAFC  2A AF AD         ld hl,(ADAF) ON ERROR address
CAFF  7C               ld a,h
CB00  B5               or l          ON ERROR address present?
CB01  E1              pop hl
CB02  C2 94 CA        jp nz,CA94     perform ERROR <e> routine
CB05  AF              xor a          a=0
CB06  CD A2 C1        call C1A2      set output channel to <a>, a= old channel
CB09  F5              push af
CB0A  CD 41 C3         call C341     output text (hl) to channel
CB0D  CD 4E C3         call C34E     output 'LF to channel
CB10  F1              pop af         restore old channel
```

```
CB11  CD A2 C1      call C1A2      set output channel to <a>, a= old channel
CB14  F1            pop af
CB15  E1            pop hl
CB16  D1            pop de
CB17  C9            ret

----- print 'undefined line <line#> in <line#>
      @ E899!
CB18  CD 86 C3      call C386      set chan 0, print char <a> at a new line
CB1B  21 23 CB      ld hl,CB23     'Undefined line
CB1E  CD 48 CB      call CB48      print text (hl) and line# <de>
CB21  18 1D         jr CB40        if from RUN mode print 'in <line#>

----- 'Undefined line
CB23  55 6E 64 65 66 69 6E 65  64 20 6C 69 6E 65 20 00      'Undefined line .

----- print 'BREAK'; if RUN: 'in <line#>
      @ CB5C! CB6B!
CB33  11 4F CB      ld de,CB4F     'Break

----- print text (de); if RUN: 'in <line#>
      @ CAD9!
CB36  CD 9D C1      call C19D      set in/out channel to 0
CB39  CD 86 C3      call C386      set chan 0, print char <a> at a new line
CB3C  EB            ex de,hl
CB3D  CD 41 C3      call C341      output text (hl) to channel

----- if from RUN mode print 'in <line#>
CB40  CD D6 DD      call DDD6      get BASIC line# at PC in <hl>, =carry
CB43  D0            ret nc
CB44  EB            ex de,hl
CB45  21 55 CB      ld hl,CB55     ' in

----- print text (hl) and line# <de>
      @ CB1E!
CB48  CD 41 C3      call C341      output text (hl) to channel
CB4B  EB            ex de,hl
CB4C  C3 79 EE      jp EE79        print line#

----- 'Break
CB4F  42 72 65 61 6B 00                                     'Break.

----- ' in
CB55  20 69 6E 20 00                                        ' in .

----- command: STOP
      @ DE9D
CB5A  C0            ret nz
CB5B  E5            push hl
CB5C  CD 33 CB      call CB33      print 'BREAK'; if RUN: 'in <line#>
CB5F  E1            pop hl
CB60  CD 93 CB      call CB93      save pointers on STOP or END
CB63  18 2B         jr CB90        reset Basic

----- command: END
      @ DE31
CB65  C0            ret nz
CB66  CD 93 CB      call CB93      save pointers on STOP or END
CB69  18 1C         jr CB87

----- perform a BREAK
      @ C411  C449  C83F  CA59  D27B  D2A5  D56B  DBB0  EA08  EC82
CB6B  CD 33 CB      call CB33      print 'BREAK'; if RUN: 'in <line#>
CB6E  2A 34 AE      ld hl,(AE34)   program counter on RUN
CB71  CD B0 CB      call CBB0      save pointers if program isn't ended
```

```
CB74  18 1A       jr CB90          reset Basic

----- perform an error break
      @ DDA8
CB76  CD D6 DD    call DDD6        get BASIC line# at PC in <hl>, =carry
CB79  30 12       jr nc,CB8D       no further line
CB7B  CD AB CB    call CBAB        reset CONTinue pointer
CB7E  3A B1 AD    ld a,(ADB1)      flag ON ERROR
CB81  B7          or a
CB82  1E 13       ld e,13          RESUME missing
CB84  C2 94 CA    jp nz,CA94       perform ERROR <e> routine
CB87  CD 98 D2    call D298        command: CLOSEIN
CB8A  CD A1 D2    call D2A1        command: CLOSEOUT
CB8D  CD CB DD    call DDCB        reset BASIC program counter

----- reset Basic
CB90  C3 64 C0    jp C064          reset Basic

----- save pointers on STOP or END
      @ CB60! CB66!
CB93  EB          ex de,hl
CB94  CD D6 DD    call DDD6        get BASIC line# at PC in <hl>, =carry
CB97  EB          ex de,hl
CB98  D0          ret nc
CB99  7E          ld a,(hl)
CB9A  FE 01       cp 01            <statement end>
CB9C  28 0B       jr z,CBA9        line not ended (':)
CB9E  23          inc hl
CB9F  7E          ld a,(hl)
CBA0  23          inc hl
CBA1  B6          or (hl)
CBA2  28 07       jr z,CBAB        reset CONTinue pointer
CBA4  23          inc hl
CBA5  CD CE DD    call DDCE        set BASIC program counter to <hl>
CBA8  23          inc hl
CBA9  18 05       jr CBB0          save pointers if program isn't ended

----- reset CONTinue pointer
      @ C17D! CB7B! CBA2'
CBAB  21 00 00    ld hl,0000
CBAE  18 0C       jr CBBC

----- save pointers if program isn't ended
      @ CAA1! CB71! CBA9'
CBB0  EB          ex de,hl
CBB1  CD D6 DD    call DDD6        get BASIC line# at PC in <hl>, =carry
CBB4  D0          ret nc
CBB5  CD D2 DD    call DDD2        get BASIC program counter in <hl>
CBB8  22 AD AD    ld (ADAD),hl     save Basic PC on STOP or END
CBBB  EB          ex de,hl
CBBC  22 AB AD    ld (ADAB),hl     CONTinue pointer
CBBF  C9          ret

----- command: CONT
      @ DE17
CBC0  C0          ret nz
CBC1  2A AB AD    ld hl,(ADAB)     CONTinue pointer
CBC4  7C          ld a,h
CBC5  B5          or 1
CBC6  1E 11       ld e,11          Cannot CONTinue
CBC8  CA 94 CA    jp z,CA94        perform ERROR <e> routine
CBCB  E5          push hl
CBCC  2A AD AD     ld hl,(ADAD)    save Basic PC on STOP or END
CBCF  CD CE DD     call DDCE       set BASIC program counter to <hl>
CBD2  CD B9 BC     call BCB9       SOUND CONTINUE stopped sounds
```

```
CBD5  E1          pop hl
CBD6  C3 74 DD    jp DD74           do the RUN LOOP

----- reset ON ERROR FLAG and ADDRESS
      @ C17A!
CBD9  AF          xor a
CBDA  32 B1 AD    ld (ADB1),a       flag ON ERROR

----- reset ON ERROR ADDRESS
      @ CBF8!
CBDD  11 00 00    ld de,0000
CBE0  ED 53 AF AD ld (ADAF),de      ON ERROR address
CBE4  C9          ret

----- here: ON ERROR
      @ C7E5
CBE5  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
CBE8  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
CBEB  A0          [GOTO]
CBEC  CD E1 CE    call CEE1         get line# into <de>
CBEF  E5          push hl
CBF0  CD 9A E7     call E79A        search line# <de> from start, <hl>=addr, nc=
CBF3  22 AF AD     ld (ADAF),hl     ON ERROR address
CBF6  E1          pop hl
CBF7  C9          ret

----- command: ON ERROR
      @ DE69
CBF8  CD DD CB    call CBDD         reset ON ERROR ADDRESS
CBFB  3A B1 AD    ld a,(ADB1)       flag ON ERROR
CBFE  B7          or a
CBFF  C8          ret z             if flag not set
CC00  C3 A4 CA    jp CAA4           perform ERROR <e> routine, part 2

----- command: RESUME [<line#>] or RESUME NEXT
      @ DE91
CC03  28 14       jr z,CC19         no argument present
CC05  FE B0       cp B0             [NEXT]
CC07  28 17       jr z,CC20         command: RESUME NEXT
CC09  CD 67 E7    call E767         get address VAL into <de>
CC0C  CD 4A DD    call DD4A         CHRGOT <a>; end of statement? else syntax er
CC0F  D5          push de
CC10  CD 2B CC     call CC2B        try to resume after error break
CC13  E1          pop hl
CC14  23          inc hl
CC15  F1          pop af
CC16  C3 93 DD    jp DD93           RUN  LOOP, part 2

CC19  CD 2B CC    call CC2B         try to resume after error break
CC1C  F1          pop af
CC1D  C3 74 DD    jp DD74           do the RUN LOOP

----- command: RESUME NEXT
CC20  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
CC23  C0          ret nz
CC24  CD 2B CC    call CC2B         try to resume after error break
CC27  23          inc hl
CC28  C3 EF E8    jp E8EF           command: DATA <list of<data>>   (skip this l

----- try to resume after error break
      @ CC10! CC19! CC24!
CC2B  3A B1 AD    ld a,(ADB1)       flag ON ERROR
CC2E  B7          or a
CC2F  1E 14       ld e,14           Unexpected RESUME
CC31  CA 94 CA    jp z,CA94         perform ERROR <e> routine
```

```
CC34  AF          xor a               reset
CC35  32 AA AD    ld (ADAA),a         last Basic ERROR number
CC38  32 B1 AD    ld (ADB1),a         flag ON ERROR
CC3B  2A A6 AD    ld hl,(ADA6)        ERROR ADDRESS (addr where error occurred)
CC3E  CD CE DD    call DDCE           set BASIC program counter to <hl>
CC41  2A A8 AD    ld hl,(ADA8)        program counter on error break
CC44  C9          ret

----- get error message text (hl)
      @ CAD0! CC58'
CC45  11 5B CC    ld de,CC5B          'ERROR MESSAGES
CC48  FE 1F       cp 1F               max = 30.
CC4A  D0          ret nc              unknown error
CC4B  B7          or a
CC4C  C8          ret z               unknown error
CC4D  47          ld b,a              error#
CC4E  1A          ld a,(de)           char from text
CC4F  13          inc de
CC50  B7          or a                end of text?
CC51  20 FB       jr nz,CC4E          skip over text
CC53  05          dec b
CC54  20 F8       jr nz,CC4E          skip over this entry
CC56  1A          ld a,(de)
CC57  B7          or a
CC58  28 EB       jr z,CC45           get error message text (hl)
CC5A  C9          ret

----- 'ERROR MESSAGES
CC5B  55 6E 6B 6E 6F 77 6E 20  65 72 72 6F 72 00                 'Unknown error.
CC69  55 6E 65 78 70 65 63 74  65 64 20 4E 45 58 54 00           'Unexpected NEXT.
CC79  53 79 6E 74 61 78 20 65  72 72 6F 72 00                    'Syntax error.
CC86  55 6E 65 78 70 65 63 74  65 64 20 52 45 54 55 52           'Unexpected RETUR
CC96  4E 00                                                      'N.
CC98  44 41 54 41 20 65 78 68  61 75 73 74 65 64 00              'DATA exhausted.
CCA7  49 6D 70 72 6F 70 65 72  20 61 72 67 75 6D 65 6E           'Improper argumen
CCB7  74 00                                                      't.
CCB9  4F 76 65 72 66 6C 6F 77  00                                'Overflow.
CCC2  4D 65 6D 6F 72 79 20 66  75 6C 6C 00                       'Memory full.
CCCE  4C 69 6E 65 20 64 6F 65  73 20 6E 6F 74 20 65 78           'Line does not ex
CCDE  69 73 74 00                                                'ist.
CCE2  53 75 62 73 63 72 69 70  74 20 6F 75 74 20 6F 66           'Subscript out of
CCF2  20 72 61 6E 67 65 00                                       ' range.
CCF9  41 72 72 61 79 20 61 6C  72 65 61 64 79 20 64 69           'Array already di
CD09  6D 65 6E 73 69 6F 6E 65  64 00                             'mensioned.
CD13  44 69 76 69 73 69 6F 6E  20 62 79 20 7A 65 72 6F           'Division by zero
CD23  00                                                         '.
CD24  49 6E 76 61 6C 69 64 20  64 69 72 65 63 74 20 63           'Invalid direct c
CD34  6F 6D 6D 61 6E 64 00                                       'ommand.
CD3B  54 79 70 65 20 6D 69 73  6D 61 74 63 68 00                 'Type mismatch.
CD49  53 74 72 69 6E 67 20 73  70 61 63 65 20 66 75 6C           'String space ful
CD59  6C 00                                                      'l.
CD5B  53 74 72 69 6E 67 20 74  6F 6F 20 6C 6F 6E 67 00           'String too long.
CD6B  53 74 72 69 6E 67 20 65  78 70 72 65 73 73 69 6F           'String expressio
CD7B  6E 20 74 6F 6F 20 63 6F  6D 70 6C 65 78 00                 'n too complex.
CD89  43 61 6E 6E 6F 74 20 43  4F 4E 54 69 6E 75 65 00           'Cannot CONTinue.
CD99  55 6E 6B 6E 6F 77 6E 20  75 73 65 72 20 66 75 6E           'Unknown user fun
CDA9  63 74 69 6F 6E 00                                          'ction.
CDAF  52 45 53 55 4D 45 20 6D  69 73 73 69 6E 67 00              'RESUME missing.
CDBE  55 6E 65 78 70 65 63 74  65 64 20 52 45 53 55 4D           'Unexpected RESUM
CDCE  45 00                                                      'E.
CDD0  44 69 72 65 63 74 20 63  6F 6D 6D 61 6E 64 20 66           'Direct command f
CDE0  6F 75 6E 64 00                                             'ound.
CDE5  4F 70 65 72 61 6E 64 20  6D 69 73 73 69 6E 67 00           'Operand missing.
CDF5  4C 69 6E 65 20 74 6F 6F  20 6C 6F 6E 67 00                 'Line too long.
CE03  45 4F 46 20 6D 65 74 00                                    'EOF met.
```

```
CE0B  46 69 6C 65 20 74 79 70  65 20 65 72 72 6F 72 00       'File type error.
CE1B  4E 45 58 54 20 6D 69 73  73 69 6E 67 00                'NEXT missing.
CE28  46 69 6C 65 20 61 6C 72  65 61 64 79 20 6F 70 65       'File already ope
CE38  6E 00                                                  'n.
CE3A  55 6E 6B 6E 6F 77 6E 20  63 6F 6D 6D 61 6E 64 00       'Unknown command.
CE4A  57 45 4E 44 20 6D 69 73  73 69 6E 67 00                'WEND missing.
CE57  55 6E 65 78 70 65 63 74  65 64 20 57 45 4E 44 00       'Unexpected WEND.

----- get byte VAL(expression) in <de>
      @ C1FE! C7E8! D2C0! D317! D3D2! D459! D489! F167! F189! F19D  F6A1!
      @ F6AB! F8D8! F9F5! FA05! FA36!
CE67  CD 86 CE    call CE86        get integer VAL(expression) in <de>
CE6A  F5          push af
CE6B  18 08        jr CE75

----- <a>=next VAL, 0=error
      @ C330! C3E3! CA8F! D34E! D4AF! D4B7! F1F6! F9A7!
CE6D  CD 86 CE    call CE86        get integer VAL(expression) in <de>
CE70  F5          push af
CE71  7A           ld a,d
CE72  B3           or e
CE73  28 36        jr z,CEAB       Error: improper argument
CE75  7A           ld a,d
CE76  B7           or a
CE77  20 32        jr nz,CEAB      Error: improper argument
CE79  F1          pop af
CE7A  7B          ld a,e
CE7B  C9          ret

----- get integer VAL of expression, neg=error
      @ C971! C979! D864!
CE7C  CD 86 CE    call CE86        get integer VAL(expression) in <de>
CE7F  F5          push af
CE80  7A           ld a,d
CE81  17           rla
CE82  38 27        jr c,CEAB       Error: improper argument
CE84  F1          pop af
CE85  C9          ret

----- get integer VAL(expression) in <de>
      @ C51A! C522! C986! CE67! CE6D! CE7C! D225! D2D7! D341! D3FF! F6D0!
CE86  CD FB CE    call CEFB        evaluate (expression), CHRGET, cp 01
CE89  F5          push af
CE8A  EB           ex de,hl
CE8B  CD 8D FE     call FE8D       function: CINT(<num expression>) in <hl>
CE8E  EB           ex de,hl
CE8F  F1          pop af
CE90  C9          ret

----- get unsigned-integer VAL(expr) in <de>
      @ D37B  EA24! EA56! EC62! EC6A! EC74! F15F! F194! F1BA! F1D2! F4F2!
CE91  CD FB CE    call CEFB        evaluate (expression), CHRGET, cp 01
CE94  F5          push af
CE95  C5           push bc
CE96  E5            push hl
CE97  CD C2 FE       call FEC2     function: UNT(<address expression>)
CE9A  EB             ex de,hl
CE9B  E1            pop hl
CE9C  C1           pop bc
CE9D  F1          pop af
CE9E  C9          ret
```

```
----- evaluate expression, release string again
      @ D273! D447! F8F8! FAC3!
CE9F  CD FB CE    call CEFB         evaluate (expression), CHRGET, cp 01
CEA2  C3 DA FB    jp FBDA           try to release string (FAC); <a>=len, z=zero

----- evaluate (string expression)
      @ F2C7! F9E9! FAB7!
CEA5  CD FB CE    call CEFB         evaluate (expression), CHRGET, cp 01
CEA8  C3 3C FF    jp FF3C           test VARTYPE for string, else error

----- Error: improper argument
      @ CE73' CE77' CE82' D0FD  D1AA  D238
CEAB  1E 05        ld e,05          Improper argument
CEAD  C3 94 CA     jp CA94          perform ERROR <e> routine

----- get line#'s, default <bc>=1, <de>=65535.
      @ E0F7! E737!
CEB0  01 01 00    ld bc,0001        =1
CEB3  11 FF FF    ld de,FFFF        =65535.
CEB6  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
CEB9  D4 51 DD    call nc,DD51      CHRGOT <a>; end of statement? =carry
CEBC  D8          ret c             return with default values
CEBD  FE 23       cp 23             '#
CEBF  C8          ret z
CEC0  FE F5       cp F5             [-]
CEC2  28 0A       jr z,CECE         only one argument
CEC4  CD E1 CE    call CEE1         get line# into <de>
CEC7  42          ld b,d
CEC8  4B          ld c,e            bc=de
CEC9  C8          ret z
CECA  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
CECD  D8          ret c
CECE  CD 37 DD    call DD37         get next argument
CED1  F5          [-]
CED2  11 FF FF    ld de,FFFF        65535.
CED5  C8          ret z
CED6  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
CED9  D8          ret c
CEDA  CD E1 CE    call CEE1         get line# into <de>
CEDD  C4 55 DD    call nz,DD55      CHRBACK comma?; if=:CHRGET <a>, scf
CEE0  C9          ret

----- get line# into <de>
      @ C052! C0E6! C0F0! C7FF! CBEC! CEC4! CEDA! DCDB! E7E6! E7F4! E7FE!
CEE1  7E          ld a,(hl)
CEE2  23          inc hl
CEE3  5E          ld e,(hl)
CEE4  23          inc hl            de = line#
CEE5  56          ld d,(hl)
CEE6  FE 1E       cp 1E             line# next?
CEE8  28 0E       jr z,CEF8         yes
CEEA  FE 1D       cp 1D             line address next?
CEEC  C2 7B D0    jp nz,D07B        Error: Syntax error
CEEF  E5          push hl
CEF0  EB           ex de,hl
CEF1  23           inc hl           hl = line address
CEF2  23           inc hl
CEF3  23           inc hl
CEF4  5E           ld e,(hl)        get number of this line
CEF5  23           inc hl
CEF6  56           ld d,(hl)
CEF7  E1          pop hl
CEF8  C3 3F DD    jp DD3F           CHRGET <a>, skip blank, cp 01
```

```
----- evaluate (expression), CHRGET, cp 01
      @ C5B0! C6C7! C7A1! CE86! CE91! CE9F! CEA5! D070! D157! D1F0! D1FB!
      @ D219! D55B! D58C! F233! F2D7! F300! F484! F8CE! FA3E! FAA1!
CEFB  C5          push bc
CEFC  2B           dec hl
CEFD  06 00        ld b,00         priority of operation
CEFF  CD 07 CF     call CF07       evaluate (expression); <b> priority
CF02  C1          pop bc
CF03  2B          dec hl
CF04  C3 3F DD    jp DD3F          CHRGET <a>, skip blank, cp 01

----- evaluate (expression); <b> priority
      @ CEFF! CF49! CF72! CFBC! CFC5!
CF07  C5          push bc
CF08  CD CB CF     call CFCB       get next char, skip blank, cp 01
CF0B  E5           push hl
CF0C  E1           pop hl
CF0D  C1          pop bc
CF0E  7E          ld a,(hl)        get operator from Basic text
CF0F  FE EE       cp EE            [>]
CF11  D8          ret c            not an operator
CF12  FE FE       cp FE            [NOT]
CF14  D0          ret nc
CF15  FE F4       cp F4            [+]
CF17  38 40       jr c,CF59        it is a relational operator
CF19  CC 45 FF    call z,FF45      get VARTYPE <a>; cp string
CF1C  20 12       jr nz,CF30       numeric expression
CF1E  C5          push bc
CF1F  E5           push hl
CF20  2A C2 B0      ld hl,(B0C2)   Floating point ACU, FAC
CF23  E3            ex (sp),hl
CF24  CD CB CF      call CFCB      get next char, skip blank, cp 01
CF27  CD 3C FF      call FF3C      test VARTYPE for string, else error
CF2A  E3            ex (sp),hl
CF2B  CD 63 F8      call F863      append string (hl) to string (FAC)
CF2E  18 DC         jr CF0C        perform next argument

----- numeric expression
CF30  7E          ld a,(hl)        get the operator
CF31  D6 F4       sub F4           [+]
CF33  87          add a,a
CF34  87          add a,a
CF35  C6 81       add a,81
CF37  5F          ld e,a
CF38  CE CF       adc a,CF         <de>=<a>*4+CF81
CF3A  93          sub e
CF3B  57          ld d,a
CF3C  EB          ex de,hl
CF3D  78          ld a,b
CF3E  BE          cp (hl)
CF3F  EB          ex de,hl
CF40  D0          ret nc
CF41  C5          push bc
CF42  CD 53 FF     call FF53       get VARTYPE, copy FAC to BASIC STACK
CF45  D5           push de
CF46  C5            push bc
CF47  1A             ld a,(de)     get old priority
CF48  47             ld b,a
CF49  CD 07 CF       call CF07     evaluate (expression); <b> priority
CF4C  C1            pop bc
CF4D  E3            ex (sp),hl
CF4E  23            inc hl
CF4F  EB            ex de,hl
CF50  79            ld a,c
CF51  CD A0 F5      call F5A0      decrement BASIC STACK pointer by <a>
```

```
CF54  CD FB FF     call FFFB       jp(de)
CF57  18 B3        jr CF0C         evaluate next element

----- it is a relational operator
CF59  78          ld a,b
CF5A  FE 0A       cp 0A            priority of operation
CF5C  D0          ret nc           it's not a relation, it's a calculation
CF5D  C5          push bc
CF5E  7E           ld a,(hl)       get operator
CF5F  D6 ED        sub ED          transform [>] =1 ...
CF61  47           ld b,a
CF62  CD 45 FF     call FF45       get VARTYPE <a>; cp string
CF65  11 A9 CF     ld de,CFA9      perform comparsion <hl>=<de>?
CF68  20 D8        jr nz,CF42      it's a numeric variable
CF6A  E5           push hl
CF6B  2A C2 B0      ld hl,(B0C2)   Floating point ACU, FAC
CF6E  E3            ex (sp),hl
CF6F  C5            push bc
CF70  06 0A          ld b,0A       priority
CF72  CD 07 CF       call CF07     evaluate (expression); <b> priority
CF75  C1            pop bc
CF76  E3            ex (sp),hl
CF77  C5            push bc
CF78  CD 97 F8       call F897     compare string (hl) with string (de)
CF7B  C1            pop bc
CF7C  CD AF CF      call CFAF      provide result of comparsion, -1,0,+1
CF7F  18 8B         jr CF0C        perform next argument

----- table of arithmetic functions, <priority>,<jp addr>
      @ CF38
CF81  0C          =12.             priority of operation
CF82  C3 CC FC    jp FCCC          perform [+] (plus)

CF85  0C          =12.
CF86  C3 E1 FC    jp FCE1          perform [-] (minus)

CF89  12          =18.
CF8A  C3 F5 FC    jp FCF5          perform [*] (multiply)

CF8D  12          =18.
CF8E  C3 12 FD    jp FD12          perform [/] (devide)

CF91  16          =22.
CF92  C3 F4 D4    jp D4F4          perform [^] (power)

CF95  10          =16.
CF96  C3 37 FD    jp FD37          perform [\] (devide=integer)

CF99  06          =6.
CF9A  C3 58 FD    jp FD58          perform <hl> [AND] <de>

CF9D  0E          =14.
CF9E  C3 49 FD    jp FD49          perform <hl> [MOD] <de>

CFA1  04          =4.
CFA2  C3 63 FD    jp FD63          perform <hl> [OR] <de>

CFA5  02          =2.
CFA6  C3 6D FD    jp FD6D          perform <hl> [XOR] <de>

----- perform comparsion <hl>=<de>?
      @ CF65:
CFA9  0A          ld a,(bc)
CFAA  C5          push bc
CFAB  CD 09 FD     call FD09       compare two numbers (int or real)
```

```
CFAE  C1         pop bc

----- provide result of comparsion, -1,0,+1
      @ CF7C!
CFAF  C6 01      add a,01
CFB1  8F         adc a,a
CFB2  A0         and b
CFB3  C6 FF      add a,FF
CFB5  9F         sbc a,a
CFB6  C3 05 FF   jp FF05          set FAC to (-1, 0, +1); <a> was FF,00,01

----- call if TOKEN [-] <expression>
      @ CFF5
CFB9  2B         dec hl
CFBA  06 14      ld b,14          =20.
CFBC  CD 07 CF   call CF07        evaluate (expression); <b> priority
CFBF  C3 89 FD   jp FD89          ...

----- call if TOKEN [NOT] <expression>
      @ CFFE
CFC2  2B         dec hl
CFC3  06 08      ld b,08          =8.
CFC5  CD 07 CF   call CF07        evaluate (expression); <b> priority
CFC8  C3 77 FD   jp FD77          perform [NOT] <hl>

----- get next char, skip blank, cp 01
      @ CF08! CF24!
CFCB  CD 3F DD   call DD3F        CHRGET <a>, skip blank, cp 01

----- call if TOKEN [+] <expression>
      @ CFF8
CFCE  28 1D      jr z,CFED        Error: Operand missing
CFD0  FE 0E      cp 0E            is it a variable?
CFD2  38 39      jr c,D00D        perform plus [+] VARIABLE
CFD4  FE 20      cp 20            is it a constant 0-9 or byte VAL?
CFD6  38 54      jr c,D02C        perform plus [+] constant 0-9
CFD8  FE 22      cp 22            '"
CFDA  CA CB F7   jp z,F7CB        [+]"text"; calculate len; copy temp to stack
CFDD  FE FF      cp FF            [TOKEN SWITCH]
CFDF  CA 80 D0   jp z,D080        proceed with token after SWITCH
CFE2  E5         push hl
CFE3  21 F2 CF    ld hl,CFF2      TOKEN - + ( NOT ERL FN MID$ @
CFE6  CD 93 FF    call FF93       search <a> in table(hl); hl=address
CFE9  E3          ex (sp),hl      returns to the function addr found
CFEA  C3 3F DD    jp DD3F         CHRGET <a>, skip blank, cp 01

----- Error: Operand missing
CFED  1E 16      ld e,16          Operand missing
CFEF  C3 94 CA   jp CA94          perform ERROR <e> routine

----- TOKEN - + ( NOT ERL FN MID$ @
      @ CFE3:
CFF2  08 78 D0   D078 [FIX]       Indirection: Syntax error
CFF5  F5 B9 CF   CFB9 [-]         call if TOKEN [-] <expression>
CFF8  F4 CE CF   CFCE [+]         call if TOKEN [+] <expression>
CFFB  28 70 D0   D070 '(          call if '(
CFFE  FE C2 CF   CFC2 [NOT]       call if TOKEN [NOT] <expression>
D001  E3 EE D0   D0EE [ERL]       function: ERL
D004  E4 30 D1   D130 [FN]        function: FN<name>(<list of<arguments>>)
D007  AC 4B F9   F94B [MID$]      function: MID$(<string expression>,<position
D00A  40 FA D0   D0FA '@          function @<used VARIABLE name>, =addr of ent
```

```
----- perform plus [+] VARIABLE
D00D  CD 90 D6     call D690         get variable entry
D010  30 0B        jr nc,D01D        initialise variable value
D012  FE 03        cp 03             <string VAR$>
D014  28 0F        jr z,D025         set FAC to <de>
D016  E5           push hl
D017  EB            ex de,hl
D018  CD 4B FF      call FF4B        set VARTYPE <a>, copy VARIABLE (hl) to FAC
D01B  E1           pop hl
D01C  C9           ret

----- initialise variable value
D01D  FE 03        cp 03             <string VAR$>
D01F  C2 F3 FE     jp nz,FEF3        set FAC to all zeroes
D022  11 2B D0     ld de,D02B        dummy string descriptor, zero len

----- set FAC to <de>
      @ D014'
D025  EB           ex de,hl
D026  22 C2 B0     ld (B0C2),hl      Floating point ACU, FAC
D029  EB           ex de,hl
D02A  C9           ret

----- dummy string descriptor, zero len
      @ D022:
D02B  00

----- perform plus [+] constant 0-9
      @ CFD6'
D02C  D6 0E        sub 0E            <const 0>
D02E  FE 0A        cp 0A             <10.?
D030  38 1D        jr c,D04F         set FAC to <a>, mark integer, CHRGET
D032  23           inc hl
D033  FE 0B        cp 0B             (19)
D035  28 17        jr z,D04E         next byte
D037  FE 0F        cp 0F             (1d)
D039  38 0E        jr c,D049         next 2 bytes are integer (line#)
D03B  FE 11        cp 11             (1f)
D03D  38 1A        jr c,D059         next 5 bytes are real VAL
D03F  20 3A        jr nz,D07B        Error: Syntax error
D041  3E 05        ld a,05           real
D043  CD 4B FF     call FF4B         set VARTYPE <a>, copy VARIABLE (hl) to FAC
D046  2B           dec hl
D047  18 24        jr D06D           CHRGET <a>, skip blank, cp 01

----- set FAC to VAL of next 2 bytes
D049  5E           ld e,(hl)
D04A  23           inc hl
D04B  56           ld d,(hl)
D04C  18 04        jr D052

----- set FAC to VAL of next byte
D04E  7E           ld a,(hl)

----- set FAC to <a>, mark integer, CHRGET
D04F  5F           ld e,a
D050  16 00        ld d,00           hi byte =0
D052  EB           ex de,hl
D053  CD 0D FF     call FF0D         set FAC to <hl> and mark integer
D056  EB           ex de,hl
D057  18 14        jr D06D           CHRGET <a>, skip blank, cp 01
```

```
----- get line# in <hl>
D059  5E          ld e,(hl)
D05A  23          inc hl
D05B  56          ld d,(hl)
D05C  E5          push hl
D05D  FE 0F       cp 0F             0F+0E=1D, =line address?
D05F  20 07       jr nz,D068        not an address
D061  13          inc de
D062  EB          ex de,hl
D063  23          inc hl
D064  23          inc hl
D065  5E          ld e,(hl)
D066  23          inc hl
D067  56          ld d,(hl)
D068  EB          ex de,hl
D069  CD 60 FE    call FE60         convert unsigned integer (hl) to real
D06C  E1          pop hl

----- CHRGET <a>, skip blank, cp 01
D06D  C3 3F DD    jp DD3F           CHRGET <a>, skip blank, cp 01

----- call if '(
      @ CFFB  D0A0!
D070  CD FB CE    call CEFB         evaluate (expression), CHRGET, cp 01
D073  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
D076  29          ')
D077  C9          ret

----- Indirection: Syntax error
      @ CFF2
D078  CD 0D AC    call AC0D         Indirection: Syntax error

----- Error: Syntax error
D07B  1E 02       ld e,02           Syntax error
D07D  C3 94 CA    jp CA94           perform ERROR <e> routine

----- proceed with token after SWITCH
      @ CFDF
D080  23          inc hl
D081  4E          ld c,(hl)
D082  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
D085  79          ld a,c
D086  FE 40       cp 40
D088  38 05       jr c,D08F         token < 40
D08A  FE 49       cp 49             token < 49?
D08C  DA BB D0    jp c,D0BB         call function routine, a=TOKEN
D08F  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
D092  28          '(
D093  79          ld a,c            token
D094  87          add a,a
D095  C6 1E       add a,1E          token * 2 + 1E
D097  4F          ld c,a
D098  FE 59       cp 59
D09A  30 0D       jr nc,D0A9        jp Unknown token after switch
D09C  FE 1D       cp 1D             <next ADDRESS>
D09E  38 0E       jr c,D0AE         call function routine, c=TOKEN
D0A0  CD 70 D0    call D070         call if '(
D0A3  E5          push hl
D0A4  CD AE D0    call D0AE         call function routine, c=TOKEN
D0A7  E1          pop hl
D0A8  C9          ret
```

```
----- jp Unknown token after switch
D0A9  CD 0A AC    call AC0A         Indirection: Undefined token after switch
D0AC  18 CD       jr D07B           Error: Syntax error

----- call function routine, c=TOKEN
      @ D09E' D0A4!
D0AE  E5          push hl
D0AF  06 00        ld b,00          priority
D0B1  21 90 D1     ld hl,D190       list of function routines
D0B4  09           add hl,bc
D0B5  7E           ld a,(hl)
D0B6  23           inc hl
D0B7  66           ld h,(hl)
D0B8  6F           ld l,a
D0B9  E3           ex (sp),hl
D0BA  C9           ret              to function

----- call function routine, a=TOKEN
      @ D08C
D0BB  E5          push hl
D0BC  4F           ld c,a
D0BD  06 00        ld b,00          priority
D0BF  21 4A D0     ld hl,D04A       D0CA-80= addr list of functions
D0C2  09           add hl,bc
D0C3  09           add hl,bc
D0C4  7E           ld a,(hl)
D0C5  23           inc hl
D0C6  66           ld h,(hl)
D0C7  6F           ld l,a
D0C8  E3           ex (sp),hl
D0C9  C9           ret              to function

----- list of function routines
D0CA  17 C4       C417              function: EOF
D0CC  DC D0       D0DC              function: ERR
D0CE  F4 D0       D0F4              function: HIMEM
D0D0  24 FA       FA24              function: INKEY$
D0D2  DB D4       D4DB              function: PI
D0D4  84 D5       D584              function: RND [(<argument>)]
D0D6  E5 D0       D0E5              function: TIME
D0D8  07 D1       D107              function: XPOS
D0DA  0E D1       D10E              function: YPOS

----- function: ERR
      @ D0CC:
D0DC  E5          push hl
D0DD  3A AA AD     ld a,(ADAA)      last Basic ERROR number
D0E0  CD 0A FF     call FF0A        set FAC to <a> and mark integer
D0E3  E1          pop hl
D0E4  C9          ret

----- function: TIME
      @ D0D6:
D0E5  E5          push hl
D0E6  CD 0D BD     call BD0D        KL TIME PLEASE in <de,hl>
D0E9  CD 7C FE     call FE7C        set FAC to <hl,de> and normalise to real
D0EC  E1          pop hl
D0ED  C9          ret

----- function: ERL
      @ D001
D0EE  E5          push hl
D0EF  CD DF CA     call CADF        get next line after error <hl>, if none <hl>
D0F2  18 0E        jr D102
```

```
----- function: HIMEM
      @ D0CE:
D0F4  E5          push hl
D0F5  2A 7B AE     ld hl,(AE7B)    himem for Basic pointer
D0F8  18 08        jr D102

----- function @<used VARIABLE name>, =addr of entry
      @ D00A
D0FA  CD 90 D6    call D690        get variable entry
D0FD  D2 AB CE    jp nc,CEAB       Error: improper argument
D100  E5          push hl
D101  EB           ex de,hl
D102  CD 60 FE     call FE60       convert unsigned integer (hl) to real
D105  E1          pop hl
D106  C9          ret

----- function: XPOS
      @ D0D8:
D107  E5          push hl
D108  CD C6 BB     call BBC6       GRA ASK CURSOR, <de>=x, <hl>=y
D10B  EB           ex de,hl
D10C  18 04        jr D112

----- function: YPOS
      @ D0DA:
D10E  E5          push hl
D10F  CD C6 BB     call BBC6       GRA ASK CURSOR, <de>=x, <hl>=y
D112  CD 0D FF     call FF0D       set FAC to <hl> and mark integer
D115  E1          pop hl
D116  C9          ret

----- command: DEF FN<name>[(<argument>)]=<expression [using argument]>
      @ DE1B
D117  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
D11A  E4          [FN]
D11B  EB          ex de,hl
D11C  CD D6 DD    call DDD6        get BASIC line# at PC in <hl>, =carry
D11F  EB          ex de,hl
D120  1E 0C       ld e,0C          Invalid direct command
D122  D2 94 CA    jp nc,CA94       perform ERROR <e> routine
D125  CD A2 D6    call D6A2        used by DEF FN and FN only ??
D128  EB          ex de,hl
D129  73          ld (hl),e
D12A  23          inc hl
D12B  72          ld (hl),d
D12C  EB          ex de,hl
D12D  C3 EF E8    jp E8EF          command: DATA <list of<data>>   (skip this 1

----- function: FN<name>(<list of<arguments>>)
      @ D004
D130  CD A2 D6    call D6A2        used by DEF FN and FN only ??
D133  C5          push bc
D134  E5           push hl
D135  EB            ex de,hl
D136  5E            ld e,(hl)
D137  23            inc hl
D138  56            ld d,(hl)
D139  EB            ex de,hl
D13A  7C            ld a,h
D13B  B5            or l
D13C  1E 12         ld e,12        Unknown user function
D13E  CA 94 CA      jp z,CA94      perform ERROR <e> routine
D141  CD 07 DA      call DA07      ...
D144  7E            ld a,(hl)
D145  FE 28         cp 28          '(
```

```
D147  20 2C       jr nz,D175      ...
D149  CD 3F DD    call DD3F       CHRGET <a>, skip blank, cp 01
D14C  E3          ex (sp),hl
D14D  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
D150  28          '(
D151  E3          ex (sp),hl
D152  CD 4B DA    call DA4B       ...
D155  E3          ex (sp),hl
D156  D5          push de
D157  CD FB CE     call CEFB      evaluate (expression), CHRGET, cp 01
D15A  E3           ex (sp),hl
D15B  78           ld a,b
D15C  CD 66 D6     call D666      adjust VARTYPE, copy result to variable
D15F  E1          pop hl
D160  CD 55 DD    call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
D163  30 07       jr nc,D16C      ...
D165  E3          ex (sp),hl
D166  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
D169  2C          ',
D16A  18 E6       jr D152         ...

D16C  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
D16F  29          ')
D170  E3          ex (sp),hl
D171  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
D174  29          ')
D175  CD 27 DA    call DA27       reset FN subprogramm pointers to zero len
D178  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
D17B  EF          [=]
D17C  CD FB CE    call CEFB       evaluate (expression), CHRGET, cp 01
D17F  C2 7B D0    jp nz,D07B      Error: Syntax error
D182  CD 30 DA    call DA30       ...
D185  CD 45 FF    call FF45       get VARTYPE <a>; cp string
D188  CC 49 FB    call z,FB49     ...
D18B  E1         pop hl
D18C  F1        pop af
D18D  C3 D7 FE  jp FED7           test <a>=VARTYPE? if not CINT,CREAL
```

```
----- list of function routines
      @ D0B1:
D190  BA F8       F8BA           function: BIN$(<unsigned integer>[,<digits>]
D192  EA F8       F8EA           function: DEC$(<num VAR>,<string VAR>)
D194  C4 F8       F8C4           function: HEX$(<unsigned integer>[,<digits>]
D196  A1 FA       FAA1           function: INSTR([<start >,]<string expr>,<se
D198  3C F9       F93C           function: LEFT$(<string expression>,<len>)
D19A  EE D1       D1EE           function: MAX(<list of<arguments>>)
D19C  EA D1       D1EA           function: MIN(<list of<arguments>>)
D19E  76 C2       C276           function: POS(#<device>)
D1A0  43 F9       F943           function: RIGHT$(<string expression>,<len>)
D1A2  19 D2       D219           function: ROUND(<expression>[,<digits>])
D1A4  36 FA       FA36           function: STRING$(<repeat>,<character>)
D1A6  E9 C4       C4E9           function: TESTR(<xd>,<yd>)
D1A8  EE C4       C4EE           function: TEST(<x>,<y>)
D1AA  AB CE       CEAB           Error: improper argument
D1AC  62 C2       C262           function: VPOS(#<device>)
D1AE  85 FD       FD85           function: ABS(<num expression>)
D1B0  10 FA       FA10           function: ASC(<string expression>)
D1B2  3E D5       D53E           function: ATN(<argument>)
D1B4  16 FA       FA16           function: CHR$(<byte value>)
D1B6  8D FE       FE8D           function: CINT(<num expression>) in <hl>
D1B8  34 D5       D534           function: COS(<argument>)
D1BA  EC FE       FEEC           function: CREAL(<numeric expression>)
D1BC  20 D5       D520           function: EXP(<argument>)
D1BE  E8 FD       FDE8           function: FIX(<numeric expression>)
D1C0  2D FC       FC2D           function: FRE(0), or FRE("")
D1C2  09 D4       D409           function: INKEY(<key#>) in <hl>
D1C4  6D F1       F16D           function: INP (<I/O address>)
D1C6  ED FD       FDED           function: INT(<numeric expression>)
D1C8  23 D4       D423           function: JOY(<stick#>)  in <hl>
D1CA  0A FA       FA0A           function: LEN(<string expression>)
D1CC  2A D5       D52A           function: LOG(<argument>)
D1CE  25 D5       D525           function: LOG10(<argument>)
D1D0  34 F8       F834           function: LOWER$(<string expression>)
D1D2  58 F1       F158           function: PEEK (<address>)
D1D4  9F C9       C99F           function: REMAIN(<timer>)
D1D6  02 FF       FF02           function: SGN(<numeric expression>)
D1D8  2F D5       D52F           function: SIN(<argument>)
D1DA  57 FA       FA57           function: SPACE$(<# of spaces>)
D1DC  29 D3       D329           function: SQ(<sound channel>)
D1DE  EF D4       D4EF           function: SQR(<argument>)
D1E0  1E F9       F91E           function: STR$(<numeric expression>)
D1E2  39 D5       D539           function: TAN(<argument>)
D1E4  C2 FE       FEC2           function: UNT(<address expression>)
D1E6  42 F8       F842           function: UPPER$(<string expression>)
D1E8  77 FA       FA77           function: VAL(<string expression>)

----- function: MIN(<list of<arguments>>)
      @ D19C
D1EA  06 FF       ld b,FF        priority for evaluation
D1EC  18 02       jr D1F0

----- function: MAX(<list of<arguments>>)
      @ D19A
D1EE  06 01       ld b,01        priority for evaluation
D1F0  CD FB CE    call CEFB      evaluate (expression), CHRGET, cp 01
D1F3  CD 55 DD    call DD55      CHRBACK comma?; if=:CHRGET <a>, scf
D1F6  30 1C       jr nc,D214     no more arguments
D1F8  CD 53 FF    call FF53      get VARTYPE, copy FAC to BASIC STACK
D1FB  CD FB CE    call CEFB      evaluate (expression), CHRGET, cp 01
D1FE  E5          push hl
D1FF  79           ld a,c
D200  CD A0 F5     call F5A0     decrement BASIC STACK pointer by <a>
D203  C5           push bc
```

```
D204  E5           push hl
D205  CD 09 FD       call FD09      compare two numbers (int or real)
D208  E1           pop hl
D209  C1          pop bc
D20A  B7          or a
D20B  28 04       jr z,D211       this argument does not change the result
D20D  B8          cp b
D20E  C4 4E FF    call nz,FF4E    copy VARIABLE (hl) to FAC
D211  E1         pop hl
D212  18 DF      jr D1F3          get next argument

D214  CD 37 DD   call DD37        CHRNEXT <a>, nz=Error; CHRGET
D217  29         ')
D218  C9         ret

----- function: ROUND(<expression>[,<digits>])
      @ D1A2
D219  CD FB CE   call CEFB        evaluate (expression), CHRGET, cp 01
D21C  CD 53 FF   call FF53        get VARTYPE, copy FAC to BASIC STACK
D21F  CD 55 DD   call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
D222  11 00 00   ld de,0000
D225  DC 86 CE   call c,CE86      get integer VAL(expression) in <de>
D228  CD 37 DD   call DD37        CHRNEXT <a>, nz=Error; CHRGET
D22B  29         ')
D22C  E5         push hl
D22D  D5          push de
D22E  21 27 00     ld hl,0027     more than 39. digits?
D231  19           add hl,de
D232  11 4F 00     ld de,004F     =79.
D235  CD B8 FF     call FFB8      test HL=DE? (try hl-de)
D238  D2 AB CE     jp nc,CEAB     Error: improper argument
D23B  D1          pop de
D23C  79          ld a,c
D23D  CD A0 F5    call F5A0       decrement BASIC STACK pointer by <a>
D240  43          ld b,e
D241  CD AF FD    call FDAF       ...
D244  E1         pop hl
D245  C9         ret

----- command: CAT, list filenames from TAPE
      @ DE09
D246  C0         ret nz
D247  E5         push hl
D248  CD AD D2    call D2AD       CAS in/out abandon, release I/O buffers
D24B  CD 37 F6    call F637       allocate a tape buffer for output
D24E  CD 9B BC    call BC9B       CAS CATALOG, (de)= 2k buffer to use
D251  CD 71 F6    call F671       release tape output buffer
D254  E1         pop hl
D255  C9         ret

----- command: OPENOUT <filename>
      @ DE6F  EC0C!
D256  CD 73 D2   call D273        get <filename> argument from Basic text
D259  CD 37 F6   call F637        allocate a tape buffer for output
D25C  C3 8C BC   jp BC8C          CAS OUT OPEN, (hl)=filename, <b>=len, (de)=2

----- command: OPENIN <filename>
      @ DE6D
D25F  CD 6A D2   call D26A        get <filename>, allocate buff, OPENIN
D262  FE 16      cp 16            [SPACE$]
D264  C8         ret z
D265  1E 19      ld e,19          File type error
D267  C3 94 CA   jp CA94          perform ERROR <e> routine
```

```
----- get <filename>, allocate buff, OPENIN
      @ D25F! EB92!
D26A  CD 73 D2     call D273          get <filename> argument from Basic text
D26D  CD 32 F6     call F632          allocate tape buffer for input
D270  C3 77 BC     jp BC77            CAS IN OPEN, (hl)=filename, <b>=len, (de)=2k

----- get <filename> argument from Basic text
      @ D256! D26A!
D273  CD 9F CE     call CE9F          evaluate expression, release string again
D276  E3           ex (sp),hl
D277  EB           ex de,hl
D278  CD 85 D2     call D285          check for '!' in filename
D27B  CA 6B CB     jp z,CB6B          perform a BREAK
D27E  E1           pop hl
D27F  D8           ret c
D280  1E 1B        ld e,1B            File already open
D282  C3 94 CA     jp CA94            perform ERROR <e> routine

----- check for '!' in filename
      @ D278!
D285  D5           push de
D286  0E 00         ld c,00           reset flag ENABLE prompt
D288  78            ld a,b            do not check for '! if b=0
D289  B7            or a
D28A  28 08         jr z,D294
D28C  7E            ld a,(hl)
D28D  FE 21         cp 21             '!
D28F  20 03         jr nz,D294        not in
D291  23            inc hl            skip over '!
D292  05            dec b             b= -1
D293  0D            dec c             set flag FF, disable prompt
D294  79            ld a,c
D295  C3 6B BC      jp BC6B           CAS NOISY, enable or disable prompt messages

----- command: CLOSEIN
      @ CB87! DE11  EBEC  EBF8
D298  E5           push hl
D299  CD 7A BC      call BC7A         CAS IN CLOSE
D29C  CD 6D F6      call F66D         release tape input buffer
D29F  E1           pop hl
D2A0  C9           ret

----- command: CLOSEOUT
      @ CB8A! DE13  EC9E!
D2A1  E5           push hl
D2A2  CD 8F BC      call BC8F         CAS OUT CLOSE
D2A5  CA 6B CB      jp z,CB6B         perform a BREAK
D2A8  CD 71 F6      call F671         release tape output buffer
D2AB  E1           pop hl
D2AC  C9           ret

----- CAS in/out abandon, release I/O buffers
      @ C15B! D248! E9E4! EB3B! EB8F! EC09!
D2AD  C5           push bc
D2AE  D5            push de
D2AF  E5             push hl
D2B0  CD 7D BC        call BC7D       CAS IN ABANDON
D2B3  CD 6D F6        call F66D       release tape input buffer
D2B6  CD 92 BC        call BC92       CAS OUT ABANDON
D2B9  CD 71 F6        call F671       release tape output buffer
D2BC  E1             pop hl
D2BD  D1            pop de
D2BE  C1           pop bc
D2BF  C9           ret
```

```
----- command: SOUND <stat>,<period>,<tim>,<vol>,<v-env>,<t-env>,<noise>
      @ DE99
D2C0  CD 67 CE    call CE67         get byte VAL(expression) in <de>
D2C3  32 B2 AD    ld (ADB2),a       SOUND chan-stat
D2C6  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
D2C9  2C          ',
D2CA  CD FF D3    call D3FF         get integer, >15. =error
D2CD  ED 53 B5 AD ld (ADB5),de      SOUND period
D2D1  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
D2D4  11 14 00    ld de,0014        default=20.
D2D7  DC 86 CE    call c,CE86       get integer VAL(expression) in <de>
D2DA  ED 53 B9 AD ld (ADB9),de      SOUND time
D2DE  01 0C 10    ld bc,100C        max =16., default =12.
D2E1  CD 0D D3    call D30D         default <a>=<c>; if comma, get byte <a>
D2E4  32 B8 AD    ld (ADB8),a       SOUND volume
D2E7  0E 00       ld c,00           default =0
D2E9  CD 0D D3    call D30D         default <a>=<c>; if comma, get byte <a>
D2EC  32 B3 AD    ld (ADB3),a       SOUND vol-env
D2EF  CD 0D D3    call D30D         default <a>=<c>; if comma, get byte <a>
D2F2  32 B4 AD    ld (ADB4),a       SOUND ton-env
D2F5  06 20       ld b,20           max =32.
D2F7  CD 0D D3    call D30D         default <a>=<c>; if comma, get byte <a>
D2FA  32 B7 AD    ld (ADB7),a       SOUND noise
D2FD  CD 4A DD    call DD4A         CHRGOT <a>; end of statement? else syntax er
D300  E5          push hl
D301  21 B2 AD     ld hl,ADB2       SOUND chan-stat
D304  CD AA BC     call BCAA        SOUND QUEUE, add a sound, (hl)=sound program
D307  E1          pop hl
D308  D8          ret c
D309  F1          pop af
D30A  C3 71 DD    jp DD71           get the program counter and RUN

----- default <a>=<c>; if comma, get byte <a>
      @ D2E1! D2E9! D2EF! D2F7!
D30D  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
D310  79          ld a,c
D311  D0          ret nc            no more argument
D312  7E          ld a,(hl)
D313  FE 2C       cp 2C             ',
D315  79          ld a,c
D316  C8          ret z

----- GNEXT byte VAL; cp b; ret c; syntax error
      @ D320! D371! D380! D3C2! D43F! D467! D4C8!
D317  CD 67 CE    call CE67         get byte VAL(expression) in <de>
D31A  B8          cp b
D31B  D8          ret c
D31C  18 2B       jr D349           Error: Improper argument

----- command: RELEASE <channels>
      @ DE89
D31E  06 08       ld b,08           maximum 7
D320  CD 17 D3    call D317         GNEXT byte VAL; cp b; ret c; syntax error
D323  E5          push hl
D324  CD B3 BC     call BCB3        SOUND RELEASE, <a>=channel(s)
D327  E1          pop hl
D328  C9          ret

----- function: SQ(<sound channel>)
      @ D1DC
D329  CD 8D FE    call FE8D         function: CINT(<num expression>) in <hl>
D32C  7D          ld a,l
D32D  B7          or a
D32E  1F          rra               test bit 0
D32F  38 06       jr c,D337
```

```
D331  1F          rra               test bit 1
D332  38 03       jr c,D337
D334  1F          rra               test bit 2
D335  30 12       jr nc,D349        Error: Improper argument
D337  B4          or h              highbyte or rest of <a> not 0?
D338  20 0F       jr nz,D349        Error: Improper argument
D33A  7D          ld a,l            restore argument
D33B  CD AD BC    call BCAD         SOUND CHECK for space in <a>, <a>=status
D33E  C3 0A FF    jp FF0A           set FAC to <a> and mark integer

----- get integer VAL in <de>, 0=error
      @ D385! D3CA!
D341  CD 86 CE    call CE86         get integer VAL(expression) in <de>
D344  7B          ld a,e
D345  87          add a,a
D346  9F          sbc a,a
D347  BA          cp d
D348  C8          ret z

----- Error: Improper argument
D349  1E 05       ld e,05           Improper argument
D34B  C3 94 CA    jp CA94           perform ERROR <e> routine

----- command: ENV <sequence#> [,<steps>,<step>,<pause>]
      @ DE35
D34E  CD 6D CE    call CE6D         <a>=next VAL, 0=error
D351  FE 10       cp 10             sequence# > 15. ?
D353  30 F4       jr nc,D349        Error: Improper argument
D355  F5          push af
D356  11 67 D3     ld de,D367       get the sequence arguments
D359  CD D8 D3     call D3D8        set up env sequence <steps>,<step>,<pause>
D35C  F1          pop af
D35D  E5          push hl
D35E  21 BB AD     ld hl,ADBB       envelope table address
D361  71           ld (hl),c
D362  CD BC BC     call BCBC        SOUND set AMPL ENVELOPE, <a>=env#, (hl)=data
D365  E1          pop hl
D366  C9          ret

----- get the sequence arguments
      @ D356:
D367  7E          ld a,(hl)
D368  FE EF       cp EF             [=]
D36A  20 12       jr nz,D37E        ...
D36C  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
D36F  06 10       ld b,10           maximum 15.
D371  CD 17 D3    call D317         GNEXT byte VAL; cp b; ret c; syntax error
D374  F6 80       or 80             set bit 7
D376  4F          ld c,a
D377  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
D37A  2C          ',
D37B  C3 91 CE    jp CE91           get unsigned-integer VAL(expr) in <de>

D37E  06 80       ld b,80           maximum 127.
D380  CD 17 D3    call D317         GNEXT byte VAL; cp b; ret c; syntax error
D383  18 40       jr D3C5           ...

----- command: ENT <sequence#> [,<steps>,<step>,<pause>]
      @ DE33
D385  CD 41 D3    call D341         get integer VAL in <de>, 0=error
D388  7A          ld a,d
D389  B7          or a
D38A  7B          ld a,e
D38B  28 02       jr z,D38F         ...
D38D  2F          cpl               make it positiv
```

```
D38E  3C          inc a
D38F  5F          ld e,a
D390  B7          or a
D391  28 B6       jr z,D349          Error: Improper argument
D393  FE 10       cp 10              max 15.
D395  30 B2       jr nc,D349         Error: Improper argument
D397  D5          push de
D398  11 AE D3     ld de,D3AE        get the sequence arguments
D39B  CD D8 D3     call D3D8         set up env sequence <steps>,<step>,<pause>
D39E  D1          pop de
D39F  E5          push hl
D3A0  21 BB AD     ld hl,ADBB        envelope table address
D3A3  7A           ld a,d
D3A4  E6 80        and 80            set bit 7
D3A6  B1           or c
D3A7  77           ld (hl),a
D3A8  7B           ld a,e
D3A9  CD BF BC     call BCBF         SOUND set TONE ENVELOPE, <a>=env#, (hl)=data
D3AC  E1          pop hl
D3AD  C9          ret

----- get the sequence arguments
      @ D398:
D3AE  7E          ld a,(hl)
D3AF  FE EF       cp EF              Centronics latch
D3B1  20 0D       jr nz,D3C0
D3B3  CD 3F DD    call DD3F          CHRGET <a>, skip blank, cp 01
D3B6  CD FF D3    call D3FF          get integer, >15. =error
D3B9  7A          ld a,d
D3BA  C6 F0       add a,F0           =240.
D3BC  4F          ld c,a
D3BD  43          ld b,e
D3BE  18 0E       jr D3CE

D3C0  06 F0       ld b,F0            maximum 239.
D3C2  CD 17 D3    call D317          GNEXT byte VAL; cp b; ret c; syntax error
D3C5  4F          ld c,a
D3C6  CD 37 DD    call DD37          CHRNEXT <a>, nz=Error; CHRGET
D3C9  2C          ',
D3CA  CD 41 D3    call D341          get integer VAL in <de>, 0=error
D3CD  43          ld b,e
D3CE  CD 37 DD    call DD37          CHRNEXT <a>, nz=Error; CHRGET
D3D1  2C          ',
D3D2  CD 67 CE    call CE67          get byte VAL(expression) in <de>
D3D5  57          ld d,a
D3D6  58          ld e,b
D3D7  C9          ret

----- set up env sequence <steps>,<step>,<pause>
      @ D359! D39B!
D3D8  01 00 05    ld bc,0500         max count of sequences = 5
D3DB  CD 55 DD    call DD55          CHRBACK comma?; if=:CHRGET <a>, scf
D3DE  30 1C       jr nc,D3FC         no more sequence
D3E0  D5          push de
D3E1  C5           push bc
D3E2  CD FB FF      call FFFB        jp(de)
D3E5  79            ld a,c           sequence#
D3E6  C1           pop bc
D3E7  C5           push bc
D3E8  E5            push hl
D3E9  21 BC AD        ld hl,ADBC     SOUND envelope address ??
D3EC  06 00           ld b,00        clear high part
D3EE  09              add hl,bc
D3EF  09              add hl,bc
D3F0  09              add hl,bc      step to sequence#
```

```
D3F1  77              ld (hl),a       store <step>,<steps>,<pause>
D3F2  23              inc hl
D3F3  73              ld (hl),e
D3F4  23              inc hl
D3F5  72              ld (hl),d
D3F6  E1              pop hl
D3F7  C1              pop bc
D3F8  0C              inc c
D3F9  D1              pop de          next sequence
D3FA  10 DF           djnz D3DB       is there another sequence
D3FC  C3 4A DD        jp DD4A         no more sequence

----- get integer, >15. =error
      @ D2CA! D3B6!
D3FF  CD 86 CE        call CE86       get integer VAL(expression) in <de>
D402  7A              ld a,d
D403  E6 F0           and F0          is it > 15.?
D405  C2 49 D3        jp nz,D349      Error: Improper argument
D408  C9              ret

----- function: INKEY(<key#>) in <hl>
      @ D1C2
D409  CD 8D FE        call FE8D       function: CINT(<num expression>) in <hl>
D40C  11 50 00        ld de,0050      max key# = 79.
D40F  CD B8 FF        call FFB8       test HL=DE? (try hl-de)
D412  30 22           jr nc,D436      Error: Improper argument
D414  7D              ld a,l
D415  CD 1E BB        call BB1E       KM TEST if KEY #<a> is pressed
D418  21 FF FF        ld hl,FFFF      -1 if no key pressed
D41B  28 03           jr z,D420       no key
D41D  69              ld l,c
D41E  26 00           ld h,00         clear hi part
D420  C3 0D FF        jp FF0D         set FAC to <hl> and mark integer

----- function: JOY(<stick#>)  in <hl>
      @ D1C8
D423  CD 24 BB        call BB24       KM GET JOYSTICKs 1=<h>, 2=<l>
D426  EB              ex de,hl
D427  CD 8D FE        call FE8D       function: CINT(<num expression>) in <hl>
D42A  7C              ld a,h
D42B  B5              or l
D42C  28 02           jr z,D430       stick# 0 was asked
D42E  53              ld d,e
D42F  2B              dec hl
D430  7C              ld a,h
D431  B5              or l
D432  7A              ld a,d
D433  CA 0A FF        jp z,FF0A       set FAC to <a> and mark integer

----- Error: Improper argument
D436  C3 49 D3        jp D349         Error: Improper argument

----- command: KEY <expansion code>,<string expression>
      @ DE49
D439  FE 8D           cp 8D           'F13
D43B  28 19           jr z,D456       command: KEY DEF <key#>,<repeat>[,<normal>[,
D43D  3E 20           ld a,20         maximum 31.
D43F  CD 17 D3        call D317       GNEXT byte VAL; cp b; ret c; syntax error
D442  F5              push af
D443  CD 37 DD        call DD37       CHRNEXT <a>, nz=Error; CHRGET
D446  2C              ',
D447  CD 9F CE        call CE9F       evaluate expression, release string again
D44A  48              ld c,b
D44B  F1              pop af
D44C  47              ld b,a
```

```
D44D  E5          push hl
D44E  EB           ex de,hl
D44F  CD 0F BB     call BB0F       KM SET EXPANSION string
D452  E1          pop hl
D453  30 E1       jr nc,D436       Error: Improper argument
D455  C9          ret

----- command: KEY DEF <key#>,<repeat>[,<normal>[,<shift>[,<control>]]]
D456  CD 3F DD    call DD3F        CHRGET <a>, skip blank, cp 01
D459  CD 67 CE    call CE67        get byte VAL(expression) in <de>
D45C  4F          ld c,a
D45D  FE 50       cp 50            max key# = 79.
D45F  30 D5       jr nc,D436       Error: Improper argument
D461  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
D464  2C          ',
D465  06 02       ld b,02          maximum 1
D467  CD 17 D3    call D317        GNEXT byte VAL; cp b; ret c; syntax error
D46A  1F          rra
D46B  9F          sbc a,a
D46C  47          ld b,a
D46D  C5          push bc
D46E  E5           push hl
D46F  79            ld a,c
D470  CD 39 BB      call BB39      KM SET REPEAT key# <a>, <b>=0 = not
D473  E1           pop hl
D474  C1          pop bc
D475  11 27 BB    ld de,BB27       KM SET TRANSLATE entry, <a>=key#, <b>=new tr
D478  CD 84 D4    call D484        set entry, if argument present
D47B  11 2D BB    ld de,BB2D       KM SET SHIFT entry, <a>=key#, <b>=new transl
D47E  CD 84 D4    call D484        set entry, if argument present
D481  11 33 BB    ld de,BB33       KM SET CONTROL entry, <a>=key#, <b>=new tran

----- set entry, if argument present
      @ D478! D47E!
D484  CD 55 DD    call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
D487  D0          ret nc
D488  D5          push de
D489  CD 67 CE     call CE67       get byte VAL(expression) in <de>
D48C  47           ld b,a
D48D  E3           ex (sp),hl
D48E  79           ld a,c
D48F  CD F8 FF     call FFF8       jp(hl)
D492  E1          pop hl
D493  C9          ret

----- command: SPEED
      @ DE9B
D494  FE A4       cp A4            [KEY]
D496  01 3F BB    ld bc,BB3F       KM SET DELAY key, <h>=start, <l>=rep. speed
D499  28 10       jr z,D4AB        set speed, KEY or INK
D49B  FE A2       cp A2            [INK]
D49D  01 3E BC    ld bc,BC3E       SCR SET FLASHING PERIODS <h,l>
D4A0  28 09       jr z,D4AB        set speed, KEY or INK
D4A2  FE D9       cp D9            [WRITE]
D4A4  28 1D       jr z,D4C3        set speed WRITE <speed>
D4A6  1E 02       ld e,02          Syntax error
D4A8  C3 94 CA    jp CA94          perform ERROR <e> routine

----- set speed, KEY or INK
      @ D499' D4A0'
D4AB  C5          push bc
D4AC  CD 3F DD     call DD3F       CHRGET <a>, skip blank, cp 01
D4AF  CD 6D CE     call CE6D       <a>=next VAL, 0=error
D4B2  4F           ld c,a
D4B3  CD 37 DD     call DD37       CHRNEXT <a>, nz=Error; CHRGET
```

```
D4B6  2C            ',
D4B7  CD 6D CE       call CE6D       <a>=next VAL, 0=error
D4BA  5F             ld e,a
D4BB  51             ld d,c
D4BC  C1            pop bc
D4BD  EB            ex de,hl
D4BE  CD F9 FF      call FFF9       jp(bc)
D4C1  EB            ex de,hl
D4C2  C9            ret

----- set speed WRITE <speed>
D4C3  CD 3F DD      call DD3F       CHRGET <a>, skip blank, cp 01
D4C6  06 02         ld b,02         max for argument = 1
D4C8  CD 17 D3      call D317       GNEXT byte VAL; cp b; ret c; syntax error
D4CB  E5            push hl
D4CC  21 A7 00       ld hl,00A7     ...
D4CF  3D             dec a
D4D0  3E 32          ld a,32        '2
D4D2  28 02          jr z,D4D6      speed '2
D4D4  29             add hl,hl
D4D5  0F             rrca
D4D6  CD 68 BC       call BC68      CAS SET write SPEED, <hl>=len of half a zero
D4D9  E1            pop hl
D4DA  C9            ret

----- function: PI
      @ D0D2:
D4DB  E5            push hl
D4DC  CD 19 FF       call FF19      set VARTYPE real; <a>=5
D4DF  CD 1D FF       call FF1D      get VARTYPE <c>, <hl>=FAC
D4E2  CD 76 BD       call BD76      REAL ARITH, PI (hl)
D4E5  E1            pop hl
D4E6  C9            ret

----- command: DEG
      @ DE23
D4E7  3E FF         ld a,FF         set DEG
D4E9  18 01         jr D4EC

----- command: RAD
      @ DE83
D4EB  AF            xor a           reset to RAD
D4EC  C3 73 BD      jp BD73         REAL ARITH, set DEG/RAD <a>

----- function: SQR(<argument>)
      @ D1DE
D4EF  01 79 BD      ld bc,BD79      REAL ARITH, SQR (hl) ^ 0.5
D4F2  18 16         jr D50A         perform function on argument

----- perform [^] (power)
      @ CF92
D4F4  E5            push hl
D4F5  C5             push bc
D4F6  CD EC FE        call FEEC     function: CREAL(<numeric expression>)
D4F9  EB              ex de,hl
D4FA  21 CB AD        ld hl,ADCB    FAC used by [^] (power)
D4FD  CD 3D BD        call BD3D     copy 5 bytes,(de)>(hl); ld a,(hl-1)
D500  C1             pop bc
D501  E3             ex (sp),hl
D502  79             ld a,c
D503  CD 4B FF       call FF4B      set VARTYPE <a>, copy VARIABLE (hl) to FAC
D506  D1            pop de
D507  01 7C BD      ld bc,BD7C      REAL ARITH, EXP; (hl)=(hl)^(de)
```

```
----- perform function on argument
      @ D4F2' D523' D528' D52D' D532' D537' D53C' D541'
D50A  CD 19 D5     call D519         change to real, perform function (de)
D50D  D8           ret c
D50E  CA EA CA     jp z,CAEA         Error: Division by zero
D511  FA F3 CA     jp m,CAF3         Error: Overflow
D514  1E 05        ld e,05           Improper argument
D516  C3 94 CA     jp CA94           perform ERROR <e> routine

----- change to real, perform function (de)
      @ D50A!
D519  C5           push bc
D51A  D5            push de
D51B  CD EC FE       call FEEC       function: CREAL(<numeric expression>)
D51E  D1            pop de
D51F  C9            ret              = jp(bc)

----- function: EXP(<argument>)
      @ D1BC
D520  01 85 BD     ld bc,BD85        REAL ARITH, get EXP
D523  18 E5        jr D50A           perform function on argument

----- function: LOG10(<argument>)
      @ D1CE
D525  01 82 BD     ld bc,BD82        REAL ARITH, LOG10 (hl)
D528  18 E0        jr D50A           perform function on argument

----- function: LOG(<argument>)
      @ D1CC
D52A  01 7F BD     ld bc,BD7F        REAL ARITH, LOG (hl)
D52D  18 DB        jr D50A           perform function on argument

----- function: SIN(<argument>)
      @ D1D8
D52F  01 88 BD     ld bc,BD88        REAL ARITH, SIN (hl)
D532  18 D6        jr D50A           perform function on argument

----- function: COS(<argument>)
      @ D1B8
D534  01 8B BD     ld bc,BD8B        REAL ARITH, COS (hl)
D537  18 D1        jr D50A           perform function on argument

----- function: TAN(<argument>)
      @ D1E2
D539  01 8E BD     ld bc,BD8E        REAL ARITH, TAN (hl)
D53C  18 CC        jr D50A           perform function on argument

----- function: ATN(<argument>)
      @ D1B2
D53E  01 91 BD     ld bc,BD91        REAL ARITH, ATN (hl)
D541  18 C7        jr D50A           perform function on argument

----- 'Random number seed
D543  52 61 6E 64 6F 6D 20 6E  75 6D 62 65 72 20 73 65       'Random number se
D553  65 64 20 3F 20 00                                      'ed ? .

----- command: RANDOMIZE [<start expression>]
      @ DE85
D559  28 06        jr z,D561         no arguments follow
D55B  CD FB CE     call CEFB         evaluate (expression), CHRGET, cp 01
D55E  E5           push hl
D55F  18 1B         jr D57C
```

```
D561  E5          push hl
D562  21 43 D5     ld hl,D543       'Random number seed
D565  CD 41 C3     call C341        output text (hl) to channel
D568  CD 3B CA     call CA3B        put 0 in edit buffer and read a line
D56B  D2 6B CB     jp nc,CB6B       perform a BREAK
D56E  CD 4E C3     call C34E        output 'LF to channel
D571  CD A3 EC     call ECA3        get either HEX or integer VAL
D574  30 EC        jr nc,D562       ask again
D576  CD 61 DD     call DD61        CHRSKIP <a>; skip over blank, tab, linefeed
D579  B7           or a
D57A  20 E6        jr nz,D562       ask again
D57C  CD EC FE     call FEEC        function: CREAL(<numeric expression>)
D57F  CD 9A BD     call BD9A        REAL ARITH, seed RANDOM NUMBER
D582  E1          pop hl
D583  C9          ret

----- function: RND [(<argument>)]
      @ D0D4:
D584  7E          ld a,(hl)
D585  FE 28       cp 28             '(
D587  20 1C       jr nz,D5A5        no argument present
D589  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
D58C  CD FB CE    call CEFB         evaluate (expression), CHRGET, cp 01
D58F  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
D592  29          ')
D593  E5          push hl
D594  CD EC FE     call FEEC        function: CREAL(<numeric expression>)
D597  CD 70 BD     call BD70        REAL ARITH, SGN (hl); <a>=FF,00,01
D59A  20 05        jr nz,D5A1       randomize if negative
D59C  CD A0 BD     call BDA0        REAL ARITH, get last RANDOM NUMBER (hl)
D59F  E1          pop hl
D5A0  C9          ret

D5A1  FC 9A BD     call m,BD9A      REAL ARITH, seed RANDOM NUMBER
D5A4  E1          pop hl

      @ D587'
D5A5  E5          push hl
D5A6  CD 16 FF     call FF16        ld hl,FAC; set VARTYPE real; <a>=5
D5A9  CD 9D BD     call BD9D        REAL ARITH, RANDOMIZE
D5AC  E1          pop hl
D5AD  C9          ret

----- reset all VARIABLE pointers to Basic start, clear ADD0..AE0C
      @ C191!
D5AE  CD BE D5    call D5BE         clear ADD0..AE0B to 0

----- reset all VARIABLE pointers to basic start
      @ C177! EB31  EBE9! F56E
D5B1  2A 83 AE    ld hl,(AE83)      end of BASIC program pointer
D5B4  22 85 AE    ld (AE85),hl      start of VAR table pointer
D5B7  22 87 AE    ld (AE87),hl      start of DIM'd VAR table pointer
D5BA  22 89 AE    ld (AE89),hl      upper end of DIM'd variables pointer
D5BD  C9          ret

----- clear ADD0..AE0B to 0
      @ D5AE!
D5BE  21 D0 AD    ld hl,ADD0        ...
D5C1  3E 36       ld a,36           count
D5C3  CD CB D5    call D5CB         clear <a> locations, starting (hl)
```

```
----- clear AE06..AE0B to 0
      @ D999!
D5C6  21 06 AE     ld hl,AE06        ...
D5C9  3E 06        ld a,06           count

----- clear <a> locations, starting (hl)
      @ D5C3! D5CF'
D5CB  36 00        ld (hl),00
D5CD  23           inc hl
D5CE  3D           dec a
D5CF  20 FA        jr nz,D5CB        clear <a> locations, starting (hl)
D5D1  C9           ret

----- clear AE04..5 to 0
      @ C186! EA77!
D5D2  21 00 00     ld hl,0000
D5D5  22 04 AE     ld (AE04),hl      ...
D5D8  C9           ret

----- <hl>=AE04       (is that all there is?)
      @ D6A8!
D5D9  3E 5B        ld a,5B

----- <hl>=2*<a>+AD4E
      @ D69A! D6BA! DA8E!
D5DB  2A 85 AE     ld hl,(AE85)      start of VAR table pointer
D5DE  2B           dec hl
D5DF  44           ld b,h
D5E0  4D           ld c,l            start of VARTABLE-1
D5E1  87           add a,a
D5E2  C6 4E        add a,4E          +AD4E
D5E4  6F           ld l,a
D5E5  CE AD        adc a,AD
D5E7  95           sub l
D5E8  67           ld h,a
D5E9  C9           ret

----- <hl>=2*<a>+AE06
      @ D7CA! D804! D8FD! D9B1! D9D3! DA9B!
D5EA  2A 87 AE     ld hl,(AE87)      start of DIM'd VAR table pointer
D5ED  2B           dec hl            bc=hl-1
D5EE  44           ld b,h
D5EF  4D           ld c,l
D5F0  E6 03        and 03            max 3.
D5F2  3D           dec a
D5F3  87           add a,a
D5F4  C6 06        add a,06
D5F6  6F           ld l,a
D5F7  CE AE        adc a,AE
D5F9  95           sub l
D5FA  67           ld h,a
D5FB  C9           ret

----- set default VARTYPE A-Z to real
      @ C194!
D5FC  01 5A 41     ld bc,415A        hex VAL for A-Z
D5FF  1E 05        ld e,05           VARTYPE real

----- predefine VARTYPE '<b>...'<c> to <a>
      @ D639!
D601  79           ld a,c
D602  90           sub b             end < start?
D603  38 3D        jr c,D642         Error: Syntax error
D605  E5           push hl
D606  3C            inc a
```

```
D607  21 CB AD     ld hl,ADCB        table of predefined VARTYPES -41
D60A  06 00        ld b,00
D60C  09           add hl,bc
D60D  73           ld (hl),e
D60E  2B           dec hl
D60F  3D           dec a
D610  20 FB        jr nz,D60D        next
D612  E1          pop hl
D613  C9          ret

----- command: DEFSTR <A[,O-Z]>
      @ DE21
D614  1E 03       ld e,03            VARTYPE string
D616  18 06       jr D61E            update predefined VARTYPE table

----- command: DEFINT <I[-N]>
      @ DE1D
D618  1E 02       ld e,02            VARTYPE integer
D61A  18 02       jr D61E            update predefined VARTYPE table

----- command: DEFREAL <B[-H]>
      @ DE1F
D61C  1E 05       ld e,05            VARTYPE real

----- update predefined VARTYPE table
D61E  7E          ld a,(hl)
D61F  CD 71 FF    call FF71          test <a> for A-Z, =carry
D622  30 1E       jr nc,D642         Error: Syntax error
D624  4F          ld c,a
D625  47          ld b,a             default end = first letter
D626  CD 3F DD    call DD3F          CHRGET <a>, skip blank, cp 01
D629  FE 2D       cp 2D              '-
D62B  20 0C       jr nz,D639         no further argument
D62D  CD 3F DD    call DD3F          CHRGET <a>, skip blank, cp 01
D630  CD 71 FF    call FF71          test <a> for A-Z, =carry
D633  30 0D       jr nc,D642         Error: Syntax error
D635  4F          ld c,a
D636  CD 3F DD    call DD3F          CHRGET <a>, skip blank, cp 01
D639  CD 01 D6    call D601          predefine VARTYPE '<b>...'<c> to <a>
D63C  CD 55 DD    call DD55          CHRBACK comma?; if=:CHRGET <a>, scf
D63F  38 DD       jr c,D61E          update predefined VARTYPE table
D641  C9          ret

----- Error: Syntax error
D642  1E 02       ld e,02            Syntax error
D644  18 06       jr D64C

----- Error: Subscript out of range
D646  1E 09       ld e,09            Subscript out of range
D648  18 02       jr D64C

----- Error: Array already dimensioned
      @ D7D0
D64A  1E 0A       ld e,0A            Array already dimensioned
D64C  C3 94 CA    jp CA94            perform ERROR <e> routine

----- EXTERNAL CALL or LET
      @ DDAC
D64F  FE F8       cp F8              '| = 7C*2
D651  CA A0 F1    jp z,F1A0          perform an EXTERNAL CALL
```

```
----- command: LET <variable>=<expression>
      @ DE4B
D654  CD 86 D6    call D686         get address of VARIABLE or subscript
D657  D5          push de
D658  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
D65B  EF           [=]
D65C  CD FB CE     call CEFB        evaluate (expression), CHRGET, cp 01
D65F  78           ld a,b
D660  E3           ex (sp),hl
D661  CD 66 D6     call D666        adjust VARTYPE, copy result to variable
D664  E1          pop hl
D665  C9          ret

----- adjust VARTYPE, copy result to variable
      @ D15C! D661! DBC3!
D666  47          ld b,a
D667  CD 23 FF    call FF23         get VARTYPE <a>;
D66A  B8          cp b
D66B  78          ld a,b
D66C  C4 D7 FE    call nz,FED7      test <a>=VARTYPE? if not CINT,CREAL

----- copy FAC to variable (hl)
      @ DB12!
D66F  CD 45 FF    call FF45         get VARTYPE <a>; cp string
D672  C2 62 FF    jp nz,FF62        copy FAC to (hl)
D675  E5          push hl
D676  CD 59 FB     call FB59        release string, allocate new one, copy strin
D679  D1          pop de
D67A  C3 66 FF    jp FF66           copy variable (hl) to (de)

----- command: DIM <name>(<maxindex.1>[,<...>][,<maxindex.n>]) [,<name>(.....)]
      @ D683' DE27
D67D  CD B5 D7    call D7B5         perform command DIM <name> (<maxindex.1>[,<.
D680  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
D683  38 F8       jr c,D67D         command: DIM <name>(<maxindex.1>[,<...>][,<m
D685  C9          ret               if no further argument

----- get address of VARIABLE or subscript
      @ C545! C9F7! D654! DB03! DB33! DCF3!
D686  CD 06 D9    call D906         NAME SEARCH; if not used before make entry
D689  CD DB D7    call D7DB         is it a subscripted variable?
D68C  38 42       jr c,D6D0         <a>=<b>=<c>=VARTYPE
D68E  18 28       jr D6B8

----- get variable entry
      @ D00D! D0FA!
D690  CD 06 D9    call D906         NAME SEARCH; if not used before make entry
D693  CD DB D7    call D7DB         is it a subscripted variable?
D696  38 38       jr c,D6D0         <a>=<b>=<c>=VARTYPE
D698  E5          push hl
D699  79           ld a,c
D69A  CD DB D5     call D5DB        <hl>=2*<a>+AD4E
D69D  CD DE D6     call D6DE        search for a FN function name
D6A0  18 2D        jr D6CF          pop; <a><b><c>=VARTYPE

----- used by DEF FN and FN only ??
      @ D125! D130!
D6A2  CD 06 D9    call D906         NAME SEARCH; if not used before make entry
D6A5  38 21       jr c,D6C8         name found, get pointer to entry (de)
D6A7  E5          push hl
D6A8  CD D9 D5     call D5D9        <hl>=AE04       (is that all there is?)
D6AB  CD DE D6     call D6DE        search for a FN function name
D6AE  D4 3D D7     call nc,D73D     ...
D6B1  18 1C        jr D6CF          pop; <a><b><c>=VARTYPE
```

```
----- used by FOR
      @ C529!
D6B3  CD 06 D9     call D906          NAME SEARCH; if not used before make entry
D6B6  38 10        jr c,D6C8          name found, get pointer to entry (de)
D6B8  E5           push hl
D6B9  79            ld a,c
D6BA  CD DB D5      call D5DB         <hl>=2*<a>+AD4E
D6BD  CD DE D6      call D6DE         search for a FN function name
D6C0  3A C1 B0      ld a,(B0C1)       VARTYPE
D6C3  D4 49 D7      call nc,D749      ...
D6C6  18 07         jr D6CF           pop; <a><b><c>=VARTYPE

----- name found, get pointer to entry (de)
      @ D6A5' D6B6'
D6C8  E5           push hl
D6C9  2A 85 AE      ld hl,(AE85)      start of VAR table pointer
D6CC  2B            dec hl
D6CD  19            add hl,de
D6CE  EB            ex de,hl
D6CF  E1           pop hl

----- <a>=<b>=<c>=VARTYPE
D6D0  3A C1 B0     ld a,(B0C1)        VARTYPE
D6D3  47           ld b,a
D6D4  4F           ld c,a
D6D5  C9           ret

----- search name, check (), a=b=c=VARTYPE
      @ DBD6!
D6D6  CD 06 D9     call D906          NAME SEARCH; if not used before make entry
D6D9  CD C1 E8     call E8C1          check if parentheses pair
D6DC  18 F2        jr D6D0            <a>=<b>=<c>=VARTYPE

----- search for a FN function name
      @ D69D! D6AB! D6BD!
D6DE  D5           push de
D6DF  EB            ex de,hl
D6E0  2A 2B AE      ld hl,(AE2B)      pointer to FN subprogram
D6E3  7C            ld a,h
D6E4  B5            or l
D6E5  28 0E         jr z,D6F5         ...
D6E7  D5            push de
D6E8  23             inc hl
D6E9  23             inc hl
D6EA  C5             push bc
D6EB  01 00 00        ld bc,0000
D6EE  CD 08 D7        call D708       ...
D6F1  C1             pop bc
D6F2  38 10          jr c,D704
D6F4  D1            pop de
D6F5  EB            ex de,hl
D6F6  E5            push hl
D6F7  CD 08 D7       call D708        ...
D6FA  38 03          jr c,D6FF
D6FC  E1            pop hl
D6FD  D1           pop de
D6FE  C9           ret

D6FF  F1            pop af
D700  E1           pop hl
D701  C3 6D D7     jp D76D            ...
```

```
D704  F1           pop af
D705  F1           pop af
D706  37           scf
D707  C9           ret

      @ D6EE! D6F7! D72F' D7CD! D807! D9D6!
D708  7E           ld a,(hl)
D709  23           inc hl
D70A  66           ld h,(hl)
D70B  6F           ld l,a
D70C  B4           or h
D70D  C8           ret z
D70E  09           add hl,bc
D70F  E5           push hl
D710  23            inc hl
D711  23            inc hl
D712  EB            ex de,hl
D713  2A 27 AE      ld hl,(AE27)     pointer to BASIC STACK
D716  1A            ld a,(de)
D717  BE            cp (hl)
D718  20 14         jr nz,D72E       ...
D71A  23            inc hl
D71B  13            inc de
D71C  17            rla
D71D  30 F7         jr nc,D716       ...
D71F  EB            ex de,hl
D720  3A C1 B0      ld a,(B0C1)      VARTYPE
D723  3D            dec a
D724  AE            xor (hl)
D725  E6 07         and 07           mask out
D727  20 05         jr nz,D72E       ...
D729  EB            ex de,hl
D72A  13            inc de
D72B  E1           pop hl
D72C  37           scf
D72D  C9           ret

D72E  E1           pop hl
D72F  18 D7        jr D708           ...

----- de=hl; skip over VARIABLE name
      @ D9A8! DAAB! DAD7!
D731  F5           push af
D732  54            ld d,h           de=hl
D733  5D            ld e,l
D734  23            inc hl
D735  23            inc hl
D736  7E            ld a,(hl)
D737  23            inc hl
D738  17            rla
D739  30 FB         jr nc,D736       next until bit 7 is found set
D73B  F1           pop af
D73C  C9           ret

      @ D6AE!
D73D  3E 02        ld a,02           <integer VAR%>
D73F  CD 49 D7     call D749         ...
D742  1B           dec de
D743  1A           ld a,(de)
D744  F6 40        or 40             '@
D746  12           ld (de),a
D747  13           inc de
D748  C9           ret
```

```
      @ D6C3! D73F!
D749  D5          push de
D74A  E5           push hl
D74B  C5            push bc
D74C  F5             push af
D74D  CD 77 D7        call D777    ...
D750  F5              push af
D751  2A 87 AE         ld hl,(AE87)start of DIM'd VAR table pointer
D754  EB               ex de,hl
D755  CD F8 F5         call F5F8   allocate space for new variables
D758  CD 3A F5         call F53A   shift DIM'd VAR pointers up by <bc>
D75B  F1              pop af
D75C  CD 8A D7        call D78A    ...
D75F  F1             pop af
D760  2B             dec hl
D761  36 00          ld (hl),00
D763  3D             dec a
D764  20 FA          jr nz,D760    ...
D766  C1            pop bc
D767  E3            ex (sp),hl
D768  CD A5 D7      call D7A5      ...
D76B  D1           pop de
D76C  E1          pop hl

      @ D701  D80F! D821!
D76D  23          inc hl
D76E  7B          ld a,e
D76F  91          sub c
D770  77          ld (hl),a
D771  23          inc hl
D772  7A          ld a,d
D773  98          sbc a,b
D774  77          ld (hl),a
D775  37          scf
D776  C9          ret

      @ D74D! D893!
D777  C6 03       add a,03         <string VAR$>
D779  4F          ld c,a
D77A  2A 27 AE    ld hl,(AE27)     pointer to BASIC STACK
D77D  06 00       ld b,00
D77F  0C          inc c
D780  04          inc b
D781  7E          ld a,(hl)
D782  23          inc hl
D783  17          rla
D784  30 F9       jr nc,D77F       ...
D786  78          ld a,b
D787  06 00       ld b,00
D789  C9          ret

      @ D75C! D89F!
D78A  62          ld h,d
D78B  6B          ld l,e
D78C  09          add hl,bc
D78D  4F          ld c,a
D78E  06 00       ld b,00
D790  E5          push hl
D791  D5           push de
D792  13            inc de
D793  13            inc de
D794  2A 27 AE      ld hl,(AE27)   pointer to BASIC STACK
D797  CD F2 FF      call FFF2      ldir
D79A  3A C1 B0      ld a,(B0C1)    VARTYPE
D79D  3D            dec a
```

```
D79E  12              ld (de),a
D79F  13              inc de
D7A0  42              ld b,d
D7A1  4B              ld c,e
D7A2  D1             pop de
D7A3  E1            pop hl
D7A4  C9            ret

      @ D768! D900! D9B4! DA62!
D7A5  7E            ld a,(hl)
D7A6  12            ld (de),a
D7A7  7B            ld a,e
D7A8  91            sub c
D7A9  77            ld (hl),a
D7AA  23            inc hl
D7AB  7E            ld a,(hl)
D7AC  F5            push af
D7AD  7A             ld a,d
D7AE  98             sbc a,b
D7AF  77             ld (hl),a
D7B0  F1            pop af
D7B1  13            inc de
D7B2  12            ld (de),a
D7B3  13            inc de
D7B4  C9            ret

----- perform command DIM <name> (<maxindex.1>[,<...>][,<...>]), <name>(....)]
      @ D67D!
D7B5  CD 06 D9      call D906         NAME SEARCH; if not used before make entry
D7B8  7E            ld a,(hl)
D7B9  FE 28         cp 28             '(
D7BB  28 05         jr z,D7C2         ok, go ahead
D7BD  EE 5B         xor 5B            '[
D7BF  C2 42 D6      jp nz,D642        Error: Syntax error
D7C2  CD 5A D8      call D85A         ...
D7C5  E5            push hl
D7C6  C5             push bc
D7C7  3A C1 B0        ld a,(B0C1)     VARTYPE
D7CA  CD EA D5        call D5EA       <hl>=2*<a>+AE06
D7CD  CD 08 D7        call D708       ...
D7D0  DA 4A D6        jp c,D64A       Error: Array already dimensioned
D7D3  C1             pop bc
D7D4  3E FF          ld a,FF          'IGNORE
D7D6  CD 8A D8       call D88A        ...
D7D9  E1            pop hl
D7DA  C9            ret

----- is it a subscripted variable?
      @ D689! D693!
D7DB  F5            push af
D7DC  7E             ld a,(hl)
D7DD  FE 28          cp 28            '(
D7DF  28 10          jr z,D7F1        it's a subscripted variable
D7E1  EE 5B          xor 5B           '[
D7E3  28 0C          jr z,D7F1        it's a subscripted variable
D7E5  F1            pop af
D7E6  D0            ret nc
D7E7  E5            push hl
D7E8  2A 85 AE       ld hl,(AE85)     start of VAR table pointer
D7EB  2B             dec hl
D7EC  19             add hl,de
D7ED  EB             ex de,hl
D7EE  E1            pop hl
D7EF  37            scf
D7F0  C9            ret
```

```
----- it's a subscripted variable
D7F1  CD 5A D8     call D85A        ...
D7F4  F1           pop af
D7F5  E5           push hl
D7F6  30 07         jr nc,D7FF      ...
D7F8  2A 87 AE      ld hl,(AE87)    start of DIM'd VAR table pointer
D7FB  2B            dec hl
D7FC  19            add hl,de
D7FD  18 15         jr D814         ...

D7FF  C5            push bc
D800  D5             push de
D801  3A C1 B0        ld a,(B0C1)   VARTYPE
D804  CD EA D5        call D5EA     <hl>=2*<a>+AE06
D807  CD 08 D7        call D708     ...
D80A  30 0F           jr nc,D81B    ...
D80C  13              inc de
D80D  13              inc de
D80E  E1             pop hl
D80F  CD 6D D7       call D76D      ...
D812  C1            pop bc
D813  EB            ex de,hl
D814  78            ld a,b
D815  96            sub (hl)
D816  C2 46 D6      jp nz,D646      Error: Subscript out of range
D819  18 0A         jr D825         ...

D81B  E1             pop hl
D81C  C1            pop bc
D81D  AF            xor a
D81E  CD 8A D8      call D88A       ...
D821  CD 6D D7      call D76D       ...
D824  EB            ex de,hl
D825  11 00 00      ld de,0000
D828  46            ld b,(hl)
D829  23            inc hl
D82A  E5            push hl
D82B  D5             push de
D82C  5E              ld e,(hl)
D82D  23              inc hl
D82E  56              ld d,(hl)
D82F  3E 02           ld a,02
D831  CD A0 F5        call F5A0     decrement BASIC STACK pointer by <a>
D834  7E              ld a,(hl)
D835  23              inc hl
D836  66              ld h,(hl)
D837  6F              ld l,a
D838  CD B8 FF        call FFB8     test HL=DE? (try hl-de)
D83B  D2 46 D6        jp nc,D646    Error: Subscript out of range
D83E  E3              ex (sp),hl
D83F  CD BE BD        call BDBE     INT ARITH, ??
D842  D1             pop de
D843  19             add hl,de
D844  EB             ex de,hl
D845  E1            pop hl
D846  23            inc hl
D847  23            inc hl
D848  05            dec b
D849  20 DF         jr nz,D82A      ...
D84B  E5            push hl
D84C  2A C1 B0       ld hl,(B0C1)   VARTYPE
D84F  26 00          ld h,00
D851  CD BE BD       call BDBE      INT ARITH, ??
D854  D1            pop de
D855  19            add hl,de
```

```
D856  EB          ex de,hl
D857  E1         pop hl
D858  37         scf
D859  C9         ret

      @ D7C2! D7F1!
D85A  D5         push de
D85B  CD 3F DD    call DD3F        CHRGET <a>, skip blank, cp 01
D85E  3A C1 B0    ld a,(B0C1)      VARTYPE
D861  F5          push af
D862  06 00        ld b,00         priority
D864  CD 7C CE     call CE7C       get integer VAL of expression, neg=error
D867  E5           push hl
D868  3E 02         ld a,02        <integer VAR%>
D86A  CD B0 F5      call F5B0      inc BASIC STACK pointer by <a>, (hl)=next lo
D86D  73            ld (hl),e
D86E  23            inc hl
D86F  72            ld (hl),d
D870  E1           pop hl
D871  04           inc b
D872  CD 55 DD     call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
D875  38 ED        jr c,D864       next dimension
D877  7E           ld a,(hl)
D878  FE 29        cp 29           ')
D87A  28 05        jr z,D881
D87C  FE 5D        cp 5D           ']
D87E  C2 42 D6     jp nz,D642      Error: Syntax error
D881  CD 3F DD     call DD3F       CHRGET <a>, skip blank, cp 01
D884  F1          pop af
D885  32 C1 B0    ld (B0C1),a      VARTYPE
D888  D1         pop de
D889  C9         ret

      @ D7D6! D81E!
D88A  E5         push hl
D88B  32 26 AE    ld (AE26),a      ...
D88E  C5          push bc
D88F  78           ld a,b
D890  87           add a,a
D891  C6 03        add a,03
D893  CD 77 D7     call D777       ...
D896  F5           push af
D897  2A 89 AE      ld hl,(AE89)   upper end of DIM'd variables pointer
D89A  EB            ex de,hl
D89B  CD F8 F5      call F5F8      allocate space for new variables
D89E  F1           pop af
D89F  CD 8A D7     call D78A       ...
D8A2  60           ld h,b
D8A3  69           ld l,c
D8A4  C1          pop bc
D8A5  D5          push de
D8A6  23           inc hl
D8A7  23           inc hl
D8A8  3A C1 B0     ld a,(B0C1)     VARTYPE
D8AB  5F           ld e,a
D8AC  16 00        ld d,00
D8AE  70           ld (hl),b
D8AF  E5           push hl
D8B0  23            inc hl

      @ D8D3'
D8B1  D5         push de
D8B2  3A 26 AE    ld a,(AE26)      ...
D8B5  B7          or a
D8B6  11 0B 00    ld de,000B
```

```
D8B9  28 0B        jr z,D8C6         ...
D8BB  E5           push hl
D8BC  3E 02         ld a,02
D8BE  CD A0 F5      call F5A0        decrement BASIC STACK pointer by <a>
D8C1  5E            ld e,(hl)
D8C2  23            inc hl
D8C3  56            ld d,(hl)
D8C4  13            inc de
D8C5  E1           pop hl
D8C6  73           ld (hl),e
D8C7  23           inc hl
D8C8  72           ld (hl),d
D8C9  23           inc hl
D8CA  E3           ex (sp),hl
D8CB  CD BE BD     call BDBE         INT ARITH, ??
D8CE  DA 46 D6     jp c,D646         Error: Subscript out of range
D8D1  EB           ex de,hl
D8D2  E1          pop hl
D8D3  10 DC       djnz D8B1          ...
D8D5  42          ld b,d
D8D6  4B          ld c,e
D8D7  54          ld d,h
D8D8  5D          ld e,l
D8D9  CD FB F5    call F5FB          ...
D8DC  22 89 AE    ld (AE89),hl       upper end of DIM'd variables pointer
D8DF  C5          push bc
D8E0  2B           dec hl
D8E1  36 00        ld (hl),00
D8E3  0B           dec bc
D8E4  78           ld a,b
D8E5  B1           or c
D8E6  20 F8        jr nz,D8E0        ...
D8E8  C1          pop bc
D8E9  E1          pop hl
D8EA  5E          ld e,(hl)
D8EB  16 00       ld d,00
D8ED  EB          ex de,hl
D8EE  29          add hl,hl
D8EF  23          inc hl
D8F0  09          add hl,bc
D8F1  EB          ex de,hl
D8F2  2B          dec hl
D8F3  2B          dec hl
D8F4  73          ld (hl),e
D8F5  23          inc hl
D8F6  72          ld (hl),d
D8F7  23          inc hl
D8F8  E3          ex (sp),hl
D8F9  EB          ex de,hl
D8FA  3A C1 B0    ld a,(B0C1)        VARTYPE
D8FD  CD EA D5    call D5EA          <hl>=2*<a>+AE06
D900  CD A5 D7    call D7A5          ...
D903  D1          pop de
D904  E1          pop hl
D905  C9          ret

----- NAME SEARCH; if not used before make entry
      @ D686! D690! D6A2! D6B3! D6D6! D7B5! D9CC!
D906  CD 7F D9    call D97F          what VARTYPE is VAR NAME?, set VARTYPE (FAC)
D909  23          inc hl
D90A  5E          ld e,(hl)
D90B  23          inc hl
D90C  56          ld d,(hl)
D90D  7A          ld a,d
D90E  B3          or e
```

```
D90F  28 0A       jr z,D91B        variable not yet used, insert pointer
D911  23          inc hl
D912  7E          ld a,(hl)
D913  17          rla              bit 7 set?
D914  30 FB       jr nc,D911       skip over name
D916  CD 3F DD    call DD3F        CHRGET <a>, skip blank, cp 01
D919  37          scf
D91A  C9          ret

----- variable not yet used, insert pointer
D91B  2B          dec hl
D91C  2B          dec hl
D91D  EB          ex de,hl
D91E  C1          pop bc
D91F  2A 27 AE    ld hl,(AE27)     pointer to BASIC STACK
D922  E5          push hl
D923  21 2B D9     ld hl,D92B      ...
D926  E5           push hl         call this routine by return
D927  C5            push bc        but first return to this routine
D928  EB             ex de,hl
D929  18 0E          jr D939       ...

      @ D923:
D92B  E5          push hl
D92C  2A 27 AE     ld hl,(AE27)    pointer to BASIC STACK
D92F  CD AC F5     call F5AC       set BASIC STACK pointer to <hl>
D932  E1          pop hl
D933  E3          ex (sp),hl
D934  22 27 AE    ld (AE27),hl     pointer to BASIC STACK
D937  E1          pop hl
D938  C9          ret

      @ D929' DA55!
D939  E5                push hl
D93A  7E                 ld a,(hl)
D93B  23                 inc hl
D93C  23                 inc hl
D93D  23                 inc hl
D93E  4E                 ld c,(hl)
D93F  CB A9              res 5,c
D941  FE 0B              cp 0B         <integer var>
D943  38 19              jr c,D95E     ...
D945  79                 ld a,c
D946  E6 1F              and 1F        mask 5 bits
D948  C6 0B              add a,0B
D94A  5F                 ld e,a
D94B  CE AE              adc a,AE      + AE0B
D94D  93                 sub e
D94E  57                 ld d,a
D94F  1A                 ld a,(de)
D950  32 C1 B0           ld (B0C1),a   VARTYPE
D953  E3                 ex (sp),hl
D954  36 0D              ld (hl),0D    <real var>
D956  FE 05              cp 05         <FAC real>
D958  28 03              jr z,D95D     ...
D95A  C6 09              add a,09      for predefined vartype
D95C  77                 ld (hl),a
D95D  E3                 ex (sp),hl
D95E  EB                 ex de,hl
D95F  3E 28              ld a,28       =40.
D961  CD B0 F5           call F5B0     inc BASIC STACK pointer by <a>, (hl)=next lo
D964  22 27 AE           ld (AE27),hl pointer to BASIC STACK
D967  06 29              ld b,29       =41.
D969  05                 dec b
D96A  CA 42 D6           jp z,D642     Error: Syntax error
```

```
D96D  1A              ld a,(de)
D96E  13              inc de
D96F  E6 DF           and DF         mask out
D971  77              ld (hl),a
D972  23              inc hl
D973  17              rla
D974  30 F3           jr nc,D969     bit 7 not set
D976  CD AC F5        call F5AC      set BASIC STACK pointer to <hl>
D979  EB              ex de,hl
D97A  2B              dec hl
D97B  D1             pop de
D97C  C3 3F DD       jp DD3F         CHRGET <a>, skip blank, cp 01

----- what VARTYPE is VAR NAME?, set VARTYPE (FAC)
      @ D906! DA52!
D97F  7E          ld a,(hl)
D980  FE 0B       cp 0B              <integer var>
D982  38 02       jr c,D986
D984  C6 F7       add a,F7           remove MARK '! % $
D986  FE 04       cp 04
D988  28 09       jr z,D993          set VARTYPE real
D98A  30 04       jr nc,D990
D98C  FE 02       cp 02              <integer VAR%>
D98E  30 05       jr nc,D995         = integer
D990  C3 42 D6    jp D642            Error: Syntax error

----- set VARTYPE real
D993  3E 05       ld a,05            <FAC real>
D995  32 C1 B0    ld (B0C1),a        VARTYPE
D998  C9          ret

      @ D9F8!
D999  CD C6 D5    call D5C6          clear AE06..AE0B to 0
D99C  2A 89 AE    ld hl,(AE89)       upper end of DIM'd variables pointer
D99F  EB          ex de,hl
D9A0  2A 87 AE    ld hl,(AE87)       start of DIM'd VAR table pointer
D9A3  CD B8 FF    call FFB8          test HL=DE? (try hl-de)
D9A6  C8          ret z              end of table
D9A7  D5          push de
D9A8  CD 31 D7     call D731         de=hl; skip over VARIABLE name
D9AB  7E           ld a,(hl)
D9AC  23           inc hl
D9AD  E6 07        and 07            mask out
D9AF  3C           inc a
D9B0  E5           push hl
D9B1  CD EA D5      call D5EA        <hl>=2*<a>+AE06
D9B4  CD A5 D7      call D7A5        ...
D9B7  E1           pop hl
D9B8  5E           ld e,(hl)
D9B9  23           inc hl
D9BA  56           ld d,(hl)
D9BB  23           inc hl
D9BC  19           add hl,de
D9BD  D1          pop de
D9BE  18 E3       jr D9A3            ...

----- command: ERASE <list of<DIM'd variable name>>
      @ DE37
D9C0  CD 89 E9    call E989          clear all VARIABLE indices
D9C3  CD CC D9    call D9CC          ERASE a <DIM'd VAR NAME>
D9C6  CD 55 DD    call DD55          CHRBACK comma?; if=:CHRGET <a>, scf
D9C9  38 F8       jr c,D9C3          erase next
D9CB  C9          ret
```

```
----- ERASE a <DIM'd VAR NAME>
      @ D9C3!
D9CC  CD 06 D9    call D906        NAME SEARCH; if not used before make entry
D9CF  E5          push hl
D9D0  3A C1 B0     ld a,(B0C1)     VARTYPE
D9D3  CD EA D5     call D5EA       <hl>=2*<a>+AE06
D9D6  CD 08 D7     call D708       ...
D9D9  E5           push hl
D9DA  EB            ex de,hl
D9DB  1E 05         ld e,05        Improper argument
D9DD  D2 94 CA      jp nc,CA94     perform ERROR <e> routine
D9E0  5E            ld e,(hl)
D9E1  23            inc hl
D9E2  56            ld d,(hl)
D9E3  23            inc hl
D9E4  19            add hl,de
D9E5  EB            ex de,hl
D9E6  2A 89 AE      ld hl,(AE89)   upper end of DIM'd variables pointer
D9E9  CD CF FF      call FFCF      HL=HL-DE
D9EC  E3            ex (sp),hl
D9ED  C1           pop bc
D9EE  EB           ex de,hl
D9EF  78           ld a,b
D9F0  B1           or c
D9F1  C4 F2 FF     call nz,FFF2    ldir
D9F4  EB           ex de,hl
D9F5  22 89 AE     ld (AE89),hl    upper end of DIM'd variables pointer
D9F8  CD 99 D9     call D999       ...
D9FB  E1          pop hl
D9FC  C9          ret

----- reset FN pointers as not used
      @ C165! CAB0!
D9FD  21 00 00    ld hl,0000
DA00  22 2B AE    ld (AE2B),hl     pointer to FN subprogram
DA03  22 29 AE    ld (AE29),hl     pointer to FN subprogram
DA06  C9          ret

      @ D141!
DA07  E5          push hl
DA08  2A 2B AE     ld hl,(AE2B)    pointer to FN subprogram
DA0B  E5           push hl
DA0C  2A 29 AE      ld hl,(AE29)   pointer to FN subprogram
DA0F  EB            ex de,hl
DA10  3E 06         ld a,06        len of a GOSUB entry
DA12  CD B0 F5      call F5B0      inc BASIC STACK pointer by <a>, (hl)=next lo
DA15  22 29 AE      ld (AE29),hl   pointer to FN subprogram
DA18  73            ld (hl),e
DA19  23            inc hl
DA1A  72            ld (hl),d
DA1B  23            inc hl
DA1C  AF            xor a
DA1D  77            ld (hl),a
DA1E  23            inc hl
DA1F  77            ld (hl),a
DA20  23            inc hl
DA21  D1           pop de
DA22  73           ld (hl),e
DA23  23           inc hl
DA24  72           ld (hl),d
DA25  E1          pop hl
DA26  C9          ret
```

```
----- reset FN subprogramm pointers to zero len
      @ D175!
DA27  E5          push hl
DA28  2A 29 AE     ld hl,(AE29)      pointer to FN subprogram
DA2B  22 2B AE     ld (AE2B),hl      pointer to FN subprogram
DA2E  E1          pop hl
DA2F  C9          ret

      @ D182!
DA30  E5          push hl
DA31  2A 29 AE     ld hl,(AE29)      pointer to FN subprogram
DA34  CD AC F5     call F5AC         set BASIC STACK pointer to <hl>
DA37  5E           ld e,(hl)
DA38  23           inc hl
DA39  56           ld d,(hl)
DA3A  23           inc hl
DA3B  EB           ex de,hl
DA3C  22 29 AE     ld (AE29),hl      pointer to FN subprogram
DA3F  EB           ex de,hl
DA40  23           inc hl
DA41  23           inc hl
DA42  5E           ld e,(hl)
DA43  23           inc hl
DA44  56           ld d,(hl)
DA45  EB           ex de,hl
DA46  22 2B AE     ld (AE2B),hl      pointer to FN subprogram
DA49  E1          pop hl
DA4A  C9          ret

      @ D152!
DA4B  E5          push hl
DA4C  3E 02        ld a,02           <integer VAR%>
DA4E  CD B0 F5     call F5B0         inc BASIC STACK pointer by <a>, (hl)=next lo
DA51  E3           ex (sp),hl
DA52  CD 7F D9     call D97F         what VARTYPE is VAR NAME?, set VARTYPE (FAC)
DA55  CD 39 D9     call D939         ...
DA58  E3           ex (sp),hl
DA59  EB           ex de,hl
DA5A  2A 29 AE     ld hl,(AE29)      pointer to FN subprogram
DA5D  23           inc hl
DA5E  23           inc hl
DA5F  01 00 00     ld bc,0000
DA62  CD A5 D7     call D7A5         ...
DA65  3A C1 B0     ld a,(B0C1)       VARTYPE
DA68  47           ld b,a
DA69  3C           inc a
DA6A  CD B0 F5     call F5B0         inc BASIC STACK pointer by <a>, (hl)=next lo
DA6D  78           ld a,b
DA6E  3D           dec a
DA6F  77           ld (hl),a
DA70  23           inc hl
DA71  EB           ex de,hl
DA72  E1          pop hl
DA73  C9          ret

      @ F778! FB1E  FC99
DA74  2A 29 AE    ld hl,(AE29)       pointer to FN subprogram
DA77  7C          ld a,h
DA78  B5          or l
DA79  28 0E       jr z,DA89          ...
DA7B  4E          ld c,(hl)
DA7C  23          inc hl
DA7D  46          ld b,(hl)
DA7E  23          inc hl
DA7F  C5          push bc
```

```
DA80  01 00 00     ld bc,0000
DA83  CD CE DA     call DACE        ...
DA86  E1          pop hl
DA87  18 EE       jr DA77          ...

DA89  01 41 1A    ld bc,1A41       ...
DA8C  C5          push bc
DA8D  79           ld a,c
DA8E  CD DB D5     call D5DB        <hl>=2*<a>+AD4E
DA91  CD CE DA     call DACE        ...
DA94  C1          pop bc
DA95  0C          inc c
DA96  05          dec b
DA97  20 F3       jr nz,DA8C       ...
DA99  3E 03       ld a,03          <string VAR$>
DA9B  CD EA D5    call D5EA        <hl>=2*<a>+AE06

      @ DACC'
DA9E  4E          ld c,(hl)
DA9F  23          inc hl
DAA0  46          ld b,(hl)
DAA1  78          ld a,b
DAA2  B1          or c
DAA3  C8          ret z
DAA4  2A 87 AE    ld hl,(AE87)     start of DIM'd VAR table pointer
DAA7  2B          dec hl
DAA8  09          add hl,bc
DAA9  E5          push hl
DAAA  D5           push de
DAAB  CD 31 D7      call D731       de=hl; skip over VARIABLE name
DAAE  D1           pop de
DAAF  23           inc hl
DAB0  4E           ld c,(hl)
DAB1  23           inc hl
DAB2  46           ld b,(hl)
DAB3  23           inc hl
DAB4  E5           push hl
DAB5  09            add hl,bc
DAB6  E3            ex (sp),hl
DAB7  4E            ld c,(hl)
DAB8  23            inc hl
DAB9  06 00         ld b,00
DABB  09            add hl,bc
DABC  09            add hl,bc
DABD  C1           pop bc
DABE  CD BE FF     call FFBE        test HL=BC? (try hl-bc)
DAC1  28 08        jr z,DACB        ...
DAC3  CD E7 DA     call DAE7        ...
DAC6  23           inc hl
DAC7  23           inc hl
DAC8  23           inc hl
DAC9  18 F3        jr DABE          ...

DACB  E1          pop hl
DACC  18 D0       jr DA9E          ...

      @ DA83! DA91! DAE5'
DACE  7E          ld a,(hl)
DACF  23          inc hl
DAD0  66          ld h,(hl)
DAD1  6F          ld l,a
DAD2  B4          or h
DAD3  C8          ret z
DAD4  09          add hl,bc
DAD5  E5          push hl
```

```
DAD6  D5          push de
DAD7  CD 31 D7     call D731      de=hl; skip over VARIABLE name
DADA  D1          pop de
DADB  7E          ld a,(hl)
DADC  23          inc hl
DADD  E6 07       and 07          mask out
DADF  FE 02       cp 02           <integer VAR%>
DAE1  CC E7 DA    call z,DAE7     ...
DAE4  E1         pop hl
DAE5  18 E7      jr DACE          ...

      @ DAC3! DAE1!
DAE7  C5         push bc
DAE8  D5          push de
DAE9  E5           push hl
DAEA  7E            ld a,(hl)
DAEB  23            inc hl
DAEC  4E            ld c,(hl)
DAED  23            inc hl
DAEE  46            ld b,(hl)
DAEF  EB            ex de,hl
DAF0  B7            or a
DAF1  C4 F8 FF      call nz,FFF8  jp(hl)
DAF4  E1           pop hl
DAF5  D1          pop de
DAF6  C1         pop bc
DAF7  C9         ret

----- command: LINE INPUT [#<device>,][;][<message>;]<string variable>
      @ DE4D
DAF8  CD 37 DD   call DD37        CHRNEXT <a>, nz=Error; CHRGET
DAFB  A3         [INPUT]
DAFC  CD CB C1   call C1CB        if '# get chan, default=0, set input channel
DAFF  F5         push af
DB00  CD 89 DB    call DB89       print message, if any; CHRGET
DB03  CD 86 D6    call D686       get address of VARIABLE or subscript
DB06  CD 3C FF    call FF3C       test VARTYPE for string, else error
DB09  E5          push hl
DB0A  D5           push de
DB0B  CD 1A DB      call DB1A     do line input here
DB0E  CD DC F7      call F7DC     test len of string; copy temp to stack
DB11  E1           pop hl
DB12  CD 6F D6     call D66F      copy FAC to variable (hl)
DB15  E1          pop hl
DB16  F1         pop af
DB17  C3 AF C1   jp C1AF          set input chan to <a>, a=old channel

----- do line input here
      @ DB0B!
DB1A  CD C0 C1   call C1C0        get input channel; cp 09
DB1D  D2 66 DC   jp nc,DC66       rad a char from input, test EOF
DB20  CD A2 C1   call C1A2        set output channel to <a>, a= old channel
DB23  F5         push af
DB24  CD AD DB    call DBAD       read a line into edit buffer
DB27  F1         pop af
DB28  C3 A2 C1   jp C1A2          set output channel to <a>, a= old channel

----- command: INPUT [#<device>,][;][<message>;]<list of<variable>>
      @ DE47
DB2B  CD CB C1   call C1CB        if '# get chan, default=0, set input channel
DB2E  F5         push af
DB2F  CD 47 DB    call DB47       do input here
DB32  D5          push de
DB33  CD 86 D6     call D686      get address of VARIABLE or subscript
DB36  E3           ex (sp),hl
```

```
DB37  3E 00        ld a,00
DB39  CD BC DB     call DBBC       ...
DB3C  E3           ex (sp),hl
DB3D  CD 55 DD     call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
DB40  38 F1        jr c,DB33       get next variable
DB42  D1           pop de
DB43  F1           pop af
DB44  C3 AF C1     jp C1AF         set input chan to <a>, a=old channel

----- do input here
      @ DB2F!
DB47  CD C0 C1     call C1C0       get input channel; cp 09
DB4A  30 3D        jr nc,DB89      print message, if any; CHRGET
DB4C  CD A2 C1     call C1A2       set output channel to <a>, a= old channel
DB4F  F5           push af
DB50  E5           push hl
DB51  CD 89 DB     call DB89       print message, if any; CHRGET
DB54  3E 3F        ld a,3F         '?
DB56  D4 56 C3     call nc,C356    output char <a> to channel
DB59  3E 20        ld a,20         'SPACE
DB5B  D4 56 C3     call nc,C356    output char <a> to channel
DB5E  E5           push hl
DB5F  CD AD DB     call DBAD       read a line into edit buffer
DB62  EB           ex de,hl
DB63  E1           pop hl
DB64  CD D3 DB     call DBD3       ...
DB67  38 09        jr c,DB72       ok
DB69  21 77 DB     ld hl,DB77      'Redo from Start
DB6C  CD 41 C3     call C341       output text (hl) to channel
DB6F  E1           pop hl
DB70  18 DE        jr DB50         try again

DB72  F1           pop af
DB73  F1           pop af
DB74  C3 A2 C1     jp C1A2         set output channel to <a>, a= old channel

----- 'Redo from Start
DB77  3F 52 65 64 6F 20 66 72  6F 6D 20 73 74 61 72 74        '?Redo from start
DB87  0A 00                                                   '..

----- print message, if any; CHRGET
      @ DB00! DB4A' DB51!
DB89  7E           ld a,(hl)
DB8A  FE 3B        cp 3B           ';
DB8C  32 2D AE     ld (AE2D),a     save for semicolon on PRINT
DB8F  CC 3F DD     call z,DD3F     CHRGET <a>, skip blank, cp 01
DB92  EE 22        xor 22          '"
DB94  C0           ret nz
DB95  CD CB F7     call F7CB       [+]"text"; calculate len; copy temp to stack
DB98  CD C0 C1     call C1C0       get input channel; cp 09
DB9B  F5           push af
DB9C  DC 28 F8     call c,F828     print this line
DB9F  F1           pop af
DBA0  D4 DA FB     call nc,FBDA    try to release string (FAC); <a>=len, z=zero
DBA3  CD 55 DD     call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
DBA6  D8           ret c
DBA7  CD 37 DD     call DD37       CHRNEXT <a>, nz=Error; CHRGET
DBAA  3B           ';
DBAB  B7           or a
DBAC  C9           ret
```

```
----- read a line into edit buffer
      @ DB24! DB5F!
DBAD  CD 3B CA    call CA3B          put 0 in edit buffer and read a line
DBB0  D2 6B CB    jp nc,CB6B         perform a BREAK
DBB3  3A 2D AE    ld a,(AE2D)        save for semicolon on PRINT
DBB6  FE 3B       cp 3B              ';
DBB8  C4 4E C3    call nz,C34E       output 'LF to channel
DBBB  C9          ret

      @ DB39! DCFA!
DBBC  D5          push de
DBBD  CD 02 DC     call DC02         ...
DBC0  30 0C        jr nc,DBCE        Error: Type mismatch
DBC2  E3           ex (sp),hl
DBC3  CD 66 D6     call D666         adjust VARTYPE, copy result to variable
DBC6  E1          pop hl
DBC7  7E          ld a,(hl)
DBC8  23          inc hl
DBC9  B7          or a
DBCA  C8          ret z
DBCB  EE 2C       xor 2C             ',
DBCD  C9          ret

DBCE  1E 0D        ld e,0D           Type mismatch
DBD0  C3 94 CA     jp CA94           perform ERROR <e> routine

      @ DB64!
DBD3  D5          push de
DBD4  E5           push hl
DBD5  D5            push de
DBD6  CD D6 D6       call D6D6       search name, check (), a=b=c=VARTYPE
DBD9  E3             ex (sp),hl
DBDA  AF             xor a
DBDB  CD 02 DC       call DC02       ...
DBDE  30 1E          jr nc,DBFE      ...
DBE0  FE 03          cp 03           <string VAR$>
DBE2  CC DA FB       call z,FBDA     try to release string (FAC); <a>=len, z=zero
DBE5  E3             ex (sp),hl
DBE6  CD 55 DD       call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
DBE9  E3             ex (sp),hl
DBEA  30 0B          jr nc,DBF7      ...
DBEC  CD 61 DD       call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
DBEF  EE 2C          xor 2C          ',
DBF1  20 0B          jr nz,DBFE      ...
DBF3  23             inc hl
DBF4  E3             ex (sp),hl
DBF5  18 DF          jr DBD6         ...

DBF7  CD 61 DD       call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
DBFA  B7             or a
DBFB  20 01          jr nz,DBFE      ...
DBFD  37             scf
DBFE  E1            pop hl
DBFF  E1           pop hl
DC00  D1          pop de
DC01  C9          ret

      @ DBBD! DBDB!
DC02  5F          ld e,a
DC03  CD 45 FF    call FF45          get VARTYPE <a>; cp string
DC06  57          ld d,a
DC07  D5          push de
DC08  20 06        jr nz,DC10        ...
DC0A  CD 21 DC     call DC21         ...
DC0D  37           scf
```

```
DC0E  18 09       jr DC19         ...

DC10  CD C0 C1    call C1C0       get input channel; cp 09
DC13  D4 38 DC    call nc,DC38    ...
DC16  CD A3 EC    call ECA3       get either HEX or integer VAL
DC19  F5          push af
DC1A  DC 61 DD    call c,DD61     CHRSKIP <a>; skip over blank, tab, linefeed
DC1D  F1          pop af
DC1E  D1          pop de
DC1F  7A          ld a,d
DC20  C9          ret

      @ DC0A!
DC21  CD C0 C1    call C1C0       get input channel; cp 09
DC24  38 06       jr c,DC2C       ...
DC26  CD 47 DC    call DC47       ...
DC29  C3 DC F7    jp F7DC         test len of string; copy temp to stack

DC2C  CD 61 DD    call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
DC2F  FE 22       cp 22           '"
DC31  CA CB F7    jp z,F7CB       [+]"text"; calculate len; copy temp to stack
DC34  7B          ld a,e
DC35  C3 E6 F7    jp F7E6         eliminate superfluous char's at string end

      @ DC13!
DC38  CD 9D DC    call DC9D       read a char, ignore 'blank, 'tab, 'LF
DC3B  30 05       jr nc,DC42      Error: EOF met
DC3D  11 C6 DC    ld de,DCC6      any of 20,09,0A,C8,2C,0D ?
DC40  18 2C       jr DC6E         ...

----- Error: EOF met
DC42  1E 18       ld e,18         EOF met
DC44  C3 94 CA    jp CA94         perform ERROR <e> routine

      @ DC26!
DC47  CD 9D DC    call DC9D       read a char, ignore 'blank, 'tab, 'LF
DC4A  30 F6       jr nc,DC42      Error: EOF met
DC4C  FE 22       cp 22           '"
DC4E  28 05       jr z,DC55       ...
DC50  11 CA DC    ld de,DCCA      is there a COMMA or 'CR
DC53  18 19       jr DC6E         ...

DC55  CD A8 DC    call DCA8       read a char, skip 'LF after 'CR
DC58  11 63 DC    ld de,DC63      ...
DC5B  38 11       jr c,DC6E       ...
DC5D  21 A4 AC    ld hl,ACA4      EDIT BUFFER
DC60  36 00       ld (hl),00      'NUL
DC62  C9          ret

      @ DC58:
DC63  FE 22       cp 22           '"
DC65  C9          ret

----- rad a char from input, test EOF
      @ DB1D
DC66  CD A8 DC    call DCA8       read a char, skip 'LF after 'CR
DC69  30 D7       jr nc,DC42      Error: EOF met
DC6B  11 CD DC    ld de,DCCD      cp 0D; ret

      @ DC40' DC53' DC5B'
DC6E  21 A4 AC    ld hl,ACA4      EDIT BUFFER
DC71  E5          push hl
DC72  06 FF       ld b,FF
DC74  CD FB FF    call FFFB       jp(de)
DC77  28 0C       jr z,DC85       ...
```

```
DC79  77           ld (hl),a
DC7A  23           inc hl
DC7B  05           dec b
DC7C  28 05        jr z,DC83        ...
DC7E  CD A8 DC     call DCA8        read a char, skip 'LF after 'CR
DC81  38 F1        jr c,DC74        ...
DC83  F6 FF        or FF            'IGNORE
DC85  36 00        ld (hl),00
DC87  E1          pop hl
DC88  C0          ret nz
DC89  FE 0D       cp 0D             'CR  (^M)
DC8B  C8          ret z
DC8C  FE 22       cp 22             '"
DC8E  C4 D0 DC    call nz,DCD0      any of 20,09,0A? =carry
DC91  C0          ret nz
DC92  CD 9D DC    call DC9D         read a char, ignore 'blank, 'tab, 'LF
DC95  D0          ret nc
DC96  CD CA DC    call DCCA         is there a COMMA or 'CR
DC99  C4 14 C4    call nz,C414      CAS RETURN, put last char read back
DC9C  C9          ret

----- read a char, ignore 'blank, 'tab, 'LF
      @ DC38! DC47! DC92! DCA4'
DC9D  CD A8 DC    call DCA8         read a char, skip 'LF after 'CR
DCA0  D0          ret nc            break
DCA1  CD D0 DC    call DCD0         any of 20,09,0A? =carry
DCA4  28 F7       jr z,DC9D         read a char, ignore 'blank, 'tab, 'LF
DCA6  37          scf
DCA7  C9          ret

----- read a char, skip 'LF after 'CR
      @ DC55! DC66! DC7E! DC9D!
DCA8  CD 24 C4    call C424         read a char from input file
DCAB  D0          ret nc
DCAC  C5          push bc
DCAD  FE 0D        cp 0D            'CR  (^M)
DCAF  06 0A        ld b,0A          'LF  (^J)
DCB1  28 05        jr z,DCB8
DCB3  B8           cp b
DCB4  20 0D        jr nz,DCC3
DCB6  06 0D        ld b,0D          'CR  (^M)
DCB8  4F           ld c,a
DCB9  CD 24 C4     call C424        read a char from input file
DCBC  30 04        jr nc,DCC2
DCBE  B8           cp b
DCBF  C4 14 C4     call nz,C414     CAS RETURN, put last char read back
DCC2  79           ld a,c
DCC3  C1          pop bc
DCC4  37          scf
DCC5  C9          ret

----- any of 20,09,0A,C8,2C,0D ?
      @ DC3D:
DCC6  CD D0 DC    call DCD0         any of 20,09,0A? =carry
DCC9  C8          ret z

----- is there a COMMA or 'CR
      @ DC50: DC96!
DCCA  FE 2C       cp 2C
DCCC  C8          ret z
```

```
----- cp 0D; ret
      @ DC6B:
DCCD  FE 0D        cp 0D             'CR  (^M)
DCCF  C9           ret

----- any of 20,09,0A? =carry
      @ DC8E! DCA1! DCC6!
DCD0  FE 20        cp 20             'SPACE
DCD2  C8           ret z
DCD3  FE 09        cp 09             'HT  (^I)
DCD5  C8           ret z
DCD6  FE 0A        cp 0A             'LF  (^J)
DCD8  C9           ret

----- command: RESTORE [<line#>]
      @ DE8F
DCD9  28 0A        jr z,DCE5         reset DATA pointer to basic start
DCDB  CD E1 CE     call CEE1         get line# into <de>
DCDE  E5           push hl
DCDF  CD 9A E7      call E79A        search line# <de> from start, <hl>=addr, nc=
DCE2  2B            dec hl
DCE3  18 2D         jr DD12          set DATA pointer to HL

----- reset DATA pointer to basic start
      @ C189  DCD9'
DCE5  E5           push hl
DCE6  2A 81 AE      ld hl,(AE81)     start of BASIC program -1 pointer
DCE9  18 27         jr DD12          set DATA pointer to HL

----- command: READ <list of<variable>>
      @ DE87
DCEB  E5           push hl
DCEC  2A 30 AE      ld hl,(AE30)     pointer to next data
DCEF  CD 17 DD      call DD17        get DATA element, z-flag if empty
DCF2  E3            ex (sp),hl
DCF3  CD 86 D6      call D686        get address of VARIABLE or subscript
DCF6  E3            ex (sp),hl
DCF7  23            inc hl
DCF8  3E 3A         ld a,3A          ':
DCFA  CD BC DB      call DBBC        ...
DCFD  2B            dec hl
DCFE  28 0B         jr z,DD0B        ...
DD00  2A 2E AE      ld hl,(AE2E)     last DATA line#
DD03  CD CE DD      call DDCE        set BASIC program counter to <hl>
DD06  1E 02         ld e,02          Syntax error
DD08  C3 94 CA      jp CA94          perform ERROR <e> routine

DD0B  E3            ex (sp),hl
DD0C  CD 55 DD      call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
DD0F  E3            ex (sp),hl
DD10  38 DD         jr c,DCEF        get next element

----- set DATA pointer to HL
      @ DCE3' DCE9'
DD12  22 30 AE      ld (AE30),hl     pointer to next data
DD15  E1           pop hl
DD16  C9           ret

----- get DATA element, z-flag if empty
      @ DCEF!
DD17  7E           ld a,(hl)
DD18  FE 2C        cp 2C
DD1A  C8           ret z
```

```
----- find next DATA
DD1B  CD EF E8    call E8EF         command: DATA <list of<data>>   (skip this 1
DD1E  B7          or a
DD1F  20 0E       jr nz,DD2F
DD21  23          inc hl
DD22  7E          ld a,(hl)
DD23  23          inc hl
DD24  B6          or (hl)
DD25  23          inc hl
DD26  1E 04       ld e,04           DATA exhausted
DD28  CA 94 CA    jp z,CA94         perform ERROR <e> routine
DD2B  22 2E AE    ld (AE2E),hl      last DATA line#
DD2E  23          inc hl
DD2F  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
DD32  FE 8C       cp 8C             [DATA]
DD34  20 E5       jr nz,DD1B        find next DATA
DD36  C9          ret

----- CHRNEXT <a>, nz=Error; CHRGET
      check, whether the next char within Basic program is equal to the char
      following the call to this routine
      @ C1F5! C22E! C280! C2ED! C32B! C49B! C4F5! C51E! C57C! C58F! C6CE!
      @ C7F2! C862! C8D2! C940! C94E! CBE8! CECE! D073! D08F! D117! D14D!
      @ D166! D16C! D171! D178! D214! D228! D2C6! D377! D3C6! D3CE! D443!
      @ D461! D4B3! D58F! D658! DAF8! DBA7! EA61! EC16! EC5E! EC66! F163!
      @ F185! F199! F2A3! F2AA! F2CA! F6A5! F8E1! F8EA! F8F1! F8FB! F94B!
      @ F96D! F993! F9A3! F9AF! F9B3! F9EC! FA01! FA3A! FA41! FAB3! FABA!
      @ FAC6!
DD37  E3          ex (sp),hl
DD38  7E          ld a,(hl)
DD39  23          inc hl
DD3A  E3          ex (sp),hl
DD3B  BE          cp (hl)
DD3C  C2 C6 DD    jp nz,DDC6        Error: syntax error

----- CHRGET <a>, skip blank, cp 01
      @ C2FD! C5AD! C7FC! C8DE  C9D3! C9F0! CA15  CA27! CA35! CA38  CBE5!
      @ CC20! CEF8  CF04  CFCB! CFEA  D06D  D082! D149! D36C! D3B3! D456!
      @ D4AC! D4C3! D589! D626! D62D! D636! D85B! D881! D916! D97C  DB8F!
      @ DD2F! DD43' DD56! DD5C! DD85! DDC0  E8BC! E943  @E9AB! EA40! F230:
      @ F25C! F2A0! F2C4! F2FA! F4B3! F6CD! F7D2
DD3F  23          inc hl
DD40  7E          ld a,(hl)
DD41  FE 20       cp 20             'SPACE
DD43  28 FA       jr z,DD3F         CHRGET <a>, skip blank, cp 01
DD45  FE 01       cp 01             end of line?
DD47  D0          ret nc
DD48  B7          or a              set z-flag
DD49  C9          ret

----- CHRGOT <a>; end of statement? else syntax error
      @ C0F3! C1F0! C5C3! CC0C! D2FD! D3FC  E0FF! E72B! E801! EA16! EA2B!
      @ EA6B! EAA9! EC3F! EC78! EC87! F1D8!
DD4A  7E          ld a,(hl)
DD4B  FE 02       cp 02             end of statement?
DD4D  D8          ret c
DD4E  C3 C6 DD    jp DDC6           Error: syntax error

----- CHRGOT <a>; end of statement? =carry
      @ C4B5! C53F! C6DB! CEB9! E9BD! F208! F21E! F2ED! F47F!
DD51  7E          ld a,(hl)
DD52  FE 02       cp 02             end of statement?
DD54  C9          ret
```

```
----- CHRBACK comma?; if=:CHRGET <a>, scf
      check, whether the last char read from the Basic program was a comma
      @ C0ED! C1ED! C240! C304! C491! C4DC! C62C! C980! C9FC! CEB6! CECA!
      @ CED6! CEDD! D160! D1F3! D21F! D2D1! D30D! D3DB! D484! D63C! D680!
      @ D872! D9C6! DB3D! DBA3! DBE6! DD0C! E7ED! E7FB! EA21! EA4E! EA5A!
      @ EC11! EC6E! F1CD! F6B2! F8D4!
DD55  2B          dec hl
DD56  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
DD59  EE 2C       xor 2C
DD5B  C0          ret nz
DD5C  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
DD5F  37          scf
DD60  C9          ret

----- CHRSKIP <a>; skip over blank, tab, linefeed
      @ C86B! D576! DBEC! DBF7! DC1A! DC2C! DD65' DD69' DD6D' E335! E6BC!
      @ E6D9! ECA8! ECE1! ED44! ED54! ED82! EDC9! EE1D! EE30!
DD61  7E          ld a,(hl)
DD62  23          inc hl
DD63  FE 20       cp 20             'SPACE
DD65  28 FA       jr z,DD61         CHRSKIP <a>; skip over blank, tab, linefeed
DD67  FE 09       cp 09             'HT  (^I)
DD69  28 F6       jr z,DD61         CHRSKIP <a>; skip over blank, tab, linefeed
DD6B  FE 0A       cp 0A             'LF  (^J)
DD6D  28 F2       jr z,DD61         CHRSKIP <a>; skip over blank, tab, linefeed
DD6F  2B          dec hl
DD70  C9          ret

----- get the program counter and RUN
      @ D30A
DD71  2A 34 AE    ld hl,(AE34)      program counter on RUN

----- do the RUN LOOP
      @ C0C9  C8B3  C8C8  CBD6  CC1D  DD8E' DDA1' DDA6'
DD74  EB          ex de,hl
DD75  2A 8B B0    ld hl,(B08B)      BASIC STACK pointer
DD78  22 32 AE    ld (AE32),hl      temp storage BASIC STACK pointer
DD7B  EB          ex de,hl
DD7C  22 34 AE    ld (AE34),hl      program counter on RUN
DD7F  CD 21 B9    call B921         KL POLL SYNCHRONOUS, check for higher priori
DD82  DC 07 C8    call c,C807       there is a higher priority
DD85  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
DD88  C4 AB DD    call nz,DDAB      look for other tokens
DD8B  7E          ld a,(hl)
DD8C  FE 01       cp 01             <statement end>
DD8E  28 E4       jr z,DD74         do the RUN LOOP
DD90  30 34       jr nc,DDC6        Error: syntax error
DD92  23          inc hl

----- RUN  LOOP, part 2
      @ C844  CAC8  CC16  E9F3
DD93  7E          ld a,(hl)         get len of this program line
DD94  23          inc hl
DD95  B6          or (hl)
DD96  23          inc hl
DD97  28 0F       jr z,DDA8         jp perform an error break
DD99  22 36 AE    ld (AE36),hl      BASIC program counter PC
DD9C  23          inc hl
DD9D  3A 38 AE    ld a,(AE38)       flag TRON/TROFF ff/0
DDA0  B7          or a
DDA1  28 D1       jr z,DD74         trace flag not set
DDA3  CD EB DD    call DDEB         trace print '[<line#>]'
DDA6  18 CC       jr DD74           do the RUN LOOP
```

```
----- jp perform an error break
DDA8  C3 76 CB    jp CB76          perform an error break

----- look for other tokens
      @ C6E5  C7F9  DD88!
DDAB  87          add a,a
DDAC  D2 4F D6    jp nc,D64F       EXTERNAL CALL or LET
DDAF  FE B9       cp B9            token > 5C?
DDB1  30 10       jr nc,DDC3       undefined TOKEN
DDB3  EB          ex de,hl
DDB4  C6 01       add a,01         calculate table offset
DDB6  6F          ld l,a
DDB7  CE DE       adc a,DE
DDB9  95          sub l
DDBA  67          ld h,a           now <hl>=<a>*2+DE01
DDBB  4E          ld c,(hl)
DDBC  23          inc hl           (bc)=called command address
DDBD  46          ld b,(hl)
DDBE  C5          push bc          this is the return address = command
DDBF  EB           ex de,hl
DDC0  C3 3F DD     jp DD3F         CHRGET <a>, skip blank, cp 01

----- undefined TOKEN
DDC3  CD 07 AC    call AC07        Indirection: Undefined token

----- Error: syntax error
DDC6  1E 02       ld e,02          Syntax error
DDC8  C3 94 CA    jp CA94          perform ERROR <e> routine

----- reset BASIC program counter
      @ C028! C096! CB8D! E102! EBF2!
DDCB  21 00 00    ld hl,0000

----- set BASIC program counter to <hl>
      @ C559! C5EE! C61B! C720! C794! C7AE! C9E9! CA0D! CAD6! CBA5! CBCF!
      @ CC3E! DD03! E90D! E920
DDCE  22 36 AE    ld (AE36),hl     BASIC program counter PC
DDD1  C9          ret

----- get BASIC program counter in <hl>
      @ C555! C5CC! C703! C75C! C788! C9C6! C9E5! CA09! CA1A! CA88! CBB5!
      @ E8FF!
DDD2  2A 36 AE    ld hl,(AE36)     BASIC program counter PC
DDD5  C9          ret

----- get BASIC line# at PC in <hl>, =carry
      @ C06D! C853! C87D! CB40! CB76! CB94! CBB1! D11C! E775! E896!
DDD6  2A 36 AE    ld hl,(AE36)     BASIC program counter PC

----- get line# at (hl) in <hl>, =carry
      @ CAE2!
DDD9  7C          ld a,h
DDDA  B5          or l
DDDB  C8          ret z            there is no line#
DDDC  7E          ld a,(hl)
DDDD  23          inc hl
DDDE  66          ld h,(hl)        ld hl,(hl)
DDDF  6F          ld l,a
DDE0  37          scf
DDE1  C9          ret
```

```
----- command: TRON
      @ DEA7
DDE2  3E FF      ld a,FF          flag set ON
DDE4  18 01      jr DDE7

----- command: TROFF
      @ C16B! DEA5
DDE6  AF         xor a            flag set OFF
DDE7  32 38 AE   ld (AE38),a      flag TRON/TROFF ff/0
DDEA  C9         ret

----- trace print '[<line#>]'
      @ DDA3!
DDEB  3E 5B      ld a,5B          '[
DDED  CD 56 C3   call C356        output char <a> to channel
DDF0  E5         push hl
DDF1  2A 36 AE    ld hl,(AE36)    BASIC program counter PC
DDF4  7E          ld a,(hl)
DDF5  23          inc hl
DDF6  66          ld h,(hl)
DDF7  6F          ld l,a
DDF8  CD 79 EE    call EE79       print line#
DDFB  E1         pop hl
DDFC  3E 5D      ld a,5D          ']
DDFE  C3 56 C3   jp C356          output char <a> to channel
```

```
----- Basic COMMAND TOKEN ADDRESS LIST
      @ DDB7
DE01  71 C9     C971        command: AFTER <time period> [,<timer>] GOSU
DE03  DF C0     C0DF        command: AUTO [<line#>][,<line step>]
DE05  21 C2     C221        command: BORDER <ink> [,<ink>]
DE07  BA F1     F1BA        command: CALL <RAM address>[,<list of<argume
DE09  46 D2     D246        command: CAT, list filenames from TAPE
DE0B  3C EA     EA3C        command: CHAIN <filename>[,<run line#>] [,DE
DE0D  32 C1     C132        command: CLEAR
DE0F  B5 C4     C4B5        command: CLG [<ink>]
DE11  98 D2     D298        command: CLOSEIN
DE13  A1 D2     D2A1        command: CLOSEOUT
DE15  5A C2     C25A        command: CLS [#<device>]
DE17  C0 CB     CBC0        command: CONT
DE19  EF E8     E8EF        command: DATA <list of<data>>   (skip this l
DE1B  17 D1     D117        command: DEF FN<name>[(<argument>)]=<express
DE1D  18 D6     D618        command: DEFINT <I[-N]>
DE1F  1C D6     D61C        command: DEFREAL <B[-H]>
DE21  14 D6     D614        command: DEFSTR <A[,O-Z]>
DE23  E7 D4     D4E7        command: DEG
DE25  28 E7     E728        command: DELETE <line#>[-<line#>]
DE27  7D D6     D67D        command: DIM <name>(<maxindex.1>[,<...>][,<m
DE29  C6 C4     C4C6        command: DRAW <x>,<y>[,<ink>]
DE2B  CB C4     C4CB        command: DRAWR <xd>,<yd>[,<ink>]
DE2D  52 C0     C052        command: EDIT <line#>
DE2F  F3 E8     E8F3        command: ELSE ' REM
DE31  65 CB     CB65        command: END
DE33  85 D3     D385        command: ENT <sequence#> [,<steps>,<step>,<p
DE35  4E D3     D34E        command: ENV <sequence#> [,<steps>,<step>,<p
DE37  C0 D9     D9C0        command: ERASE <list of<DIM'd variable name>
DE39  8F CA     CA8F        command: ERROR <error#>
DE3B  79 C9     C979        command: EVERY <time period> [,<timer>] GOSU
DE3D  29 C5     C529        command: FOR <variable> = <start> TO <end> [
DE3F  ED C6     C6ED        command: GOSUB <line#>
DE41  E8 C6     C6E8        command: GOTO <line#>
DE43  C7 C6     C6C7        command: IF <logic expr>
DE45  2A C2     C22A        command: INK<ink>,<colour>[,<colour>]
DE47  2B DB     DB2B        command: INPUT [#<device>,][;][<message>;]<l
DE49  39 D4     D439        command: KEY <expansion code>,<string expres
DE4B  54 D6     D654        command: LET <variable>=<expression>
DE4D  F8 DA     DAF8        command: LINE INPUT [#<device>,][;][<message
DE4F  F7 E0     E0F7        command: LIST [<line>[-<line>]][,#<device>]
DE51  F6 E9     E9F6        command: LOAD <filename>[,<startaddress>]
DE53  D2 C2     C2D2        command: LOCATE [#<device>,] <x coord>, <y c
DE55  EF F4     F4EF        command: MEMORY <address>
DE57  A6 EA     EAA6        command: MERGE [<filename>]
DE59  93 F9     F993        command: MID$(<stringvar>,<startpos>,<len>)=
DE5B  4F C2     C24F        command: MODE <mode>
DE5D  05 C5     C505        command: MOVE <x>, <y>
DE5F  0A C5     C50A        command: MOVER <xd>,<yd>
DE61  FB C5     C5FB        command: NEXT [<list of <variable>>]
DE63  2B C1     C12B        command: NEW
DE65  E3 C7     C7E3        command: ON
DE67  CB C8     C8CB        command: ON BREAK
DE69  F8 CB     CBF8        command: ON ERROR
DE6B  40 C9     C940        command: ON SQ(<sound channnel>) GOSUB
DE6D  5F D2     D25F        command: OPENIN <filename>
DE6F  56 D2     D256        command: OPENOUT <filename>
DE71  8C C4     C48C        command: ORIGIN <x>,<y> [,<left>,<right>,<to
DE73  77 F1     F177        command: OUT <I/O address>,<byte value>
DE75  0A C2     C20A        command: PAPER [#<device>,] <ink>
DE77  12 C2     C212        command: PEN [#<device>,] <ink>
DE79  D0 C4     C4D0        command: PLOT <x>,<y>[,<ink>]
DE7B  D5 C4     C4D5        command: PLOTR <xd>,<yd>[,<ink>]
DE7D  5F F1     F15F        command: POKE <address>,<byte value>
```

```
DE7F  FD F1       F1FD              command: PRINT [#<device>,][<list of<variabl
DE81  F3 E8       E8F3              command: ELSE ' REM
DE83  EB D4       D4EB              command: RAD
DE85  59 D5       D559              command: RANDOMIZE [<start expression>]
DE87  EB DC       DCEB              command: READ <list of<variable>>
DE89  1E D3       D31E              command: RELEASE <channels>
DE8B  F3 E8       E8F3              command: ELSE ' REM
DE8D  DF E7       E7DF              command: RENUM [<line#>][,[<old line#>][,<st
DE8F  D9 DC       DCD9              command: RESTORE [<line#>]
DE91  03 CC       CC03              command: RESUME [<line#>] or RESUME NEXT
DE93  0F C7       C70F              command: RETURN
DE95  BD E9       E9BD              command: RUN [<line#>]
DE97  09 EC       EC09              command: SAVE <filename>[,<filetype>][,<star
DE99  C0 D2       D2C0              command: SOUND <stat>,<period>,<tim>,<vol>,<
DE9B  94 D4       D494              command: SPEED
DE9D  5A CB       CB5A              command: STOP
DE9F  9D F6       F69D              command: SYMBOL <symbol#>,<list of<parameter
DEA1  19 C3       C319              command: TAG [#<device>]
DEA3  20 C3       C320              command: TAGOFF [#<device>]
DEA5  E6 DD       DDE6              command: TROFF
DEA7  E2 DD       DDE2              command: TRON
DEA9  7D F1       F17D              command: WAIT <I/O address>,<AND mask>[,<XOR
DEAB  76 C7       C776              command: WEND
DEAD  47 C7       C747              command: WHILE <logic expression>
DEAF  E3 C3       C3E3              command: WIDTH <width>
DEB1  E1 C2       C2E1              command: WINDOW [#<device>,]<left>,<right>,<
DEB3  7B F4       F47B              command: WRITE [#<device>,][<list of<variabl
DEB5  F6 F1       F1F6              command: ZONE <byte value>
DEB7  E1 C8       C8E1              command: DI
DEB9  E7 C8       C8E7              command: EI

----- assemble a program line; (hl)=edit buffer
      @ C0C2! E6D5!
DEBB  D5          push de
DEBC  EB           ex de,hl
DEBD  2A 7F AE     ld hl,(AE7F)     low memory boundary pointer
DEC0  EB           ex de,hl
DEC1  D5           push de          low mem
DEC2  AF            xor a           reset
DEC3  32 39 AE      ld (AE39),a     flag used assembling a basic line
DEC6  01 2C 01      ld bc,012C      space below basic program
DEC9  CD E1 DE      call DEE1       convert a Basic line element to Basic code
DECC  7E            ld a,(hl)       points to edit buffer
DECD  B7            or a
DECE  20 F9         jr nz,DEC9      next char from edit buffer
DED0  3E 2D         ld a,2D
DED2  91            sub c
DED3  4F            ld c,a
DED4  3E 01         ld a,01         012D-<bc>
DED6  98            sbc a,b
DED7  47            ld b,a          <bc>=len of the assembled line
DED8  AF            xor a           =0
DED9  12            ld (de),a
DEDA  13            inc de
DEDB  12            ld (de),a
DEDC  13            inc de
DEDD  12            ld (de),a       append 00 00 00, end of line/len of next
DEDE  E1           pop hl
DEDF  D1          pop de
DEE0  C9          ret
```

```
----- convert a Basic line element to Basic code
      @ DEC9!
DEE1  CD 10 AC    call AC10         Indirection: Line Assembling
DEE4  7E          ld a,(hl)         get a char from edit buffer
DEE5  B7          or a
DEE6  C8          ret z             all done, return
DEE7  CD 71 FF    call FF71         test <a> for A-Z, =carry
DEEA  38 1D       jr c,DF09         it is a VARIABLE name
DEEC  CD 7F FF    call FF7F         test <a> for '.' or digit, =CARRY
DEEF  DA FF DF    jp c,DFFF         it is a numeric value
DEF2  FE 26       cp 26             '&
DEF4  CA 5A E0    jp z,E05A         it is a &HEX value
DEF7  23          inc hl
DEF8  FE 80       cp 80
DEFA  D0          ret nc            take char's > 80 as is
DEFB  FE 20       cp 20             'SPACE
DEFD  C2 80 E0    jp nz,E080        it's neither letter nor digit nor blank
DF00  3A 00 AC    ld a,(AC00)       BASIC flag ??
DF03  B7          or a
DF04  C0          ret nz
DF05  3E 20       ld a,20           'SPACE
DF07  18 1C       jr DF25           PUT CHAR <a> into Basic line (de), inc de, d

----- it is a VARIABLE name
      @ DEEA'
DF09  CD 4E DF    call DF4E         ...
DF0C  D8          ret c
DF0D  FE C5       cp C5             [REM]
DF0F  CA ED E0    jp z,E0ED         copy REMark to basic text
DF12  E5          push hl
DF13  21 30 DF     ld hl,DF30       token: DATA DEFINT DEFSTR DEFREAL
DF16  CD AA FF     call FFAA        search <a> in table(hl); =carry
DF19  E1          pop hl
DF1A  38 19       jr c,DF35         copy edit buffer to basic text
DF1C  F5          push af
DF1D  FE 97        cp 97            [ELSE]
DF1F  3E 01        ld a,01          [ATN]
DF21  CC 25 DF     call z,DF25      PUT CHAR <a> into Basic line (de), inc de, d
DF24  F1          pop af

----- PUT CHAR <a> into Basic line (de), inc de, dec bc
      @ DF07' DF21! DF35! DF7F! DF83! DFA4! DFA8! DFAC! DFB7! E007  E00C
      @ E041! E045! E04E! E06A! E077! E0B0  E0BF! E0CA  E0CD! E0D4! E0E8!
      @ E0ED!
DF25  12          ld (de),a         store byte
DF26  13          inc de            inc pointer
DF27  0B          dec bc            dec count
DF28  79          ld a,c
DF29  B0          or b              check for zero
DF2A  C0          ret nz
DF2B  1E 17       ld e,17           Line too long
DF2D  C3 94 CA    jp CA94           perform ERROR <e> routine

----- token: DATA DEFINT DEFSTR DEFREAL
      @ DF13:
DF30  8C          [DATA]
DF31  8E          [DEFINT]
DF32  90          [DEFSTR]
DF33  8F          [DEFREAL]
DF34  00
```

```
----- copy edit buffer to basic text
      @ DF1A' DF42'
DF35  CD 25 DF   call DF25          PUT CHAR <a> into Basic line (de), inc de, d
DF38  7E         ld a,(hl)          hl = buffer pointer
DF39  B7         or a
DF3A  C8         ret z              end of buffer
DF3B  FE 3A      cp 3A              ':
DF3D  28 0A      jr z,DF49          end of statement
DF3F  23         inc hl
DF40  FE 22      cp 22              '"
DF42  20 F1      jr nz,DF35         copy edit buffer to basic text
DF44  CD BF E0   call E0BF          copy text up to '" or end of buffer
DF47  18 EF      jr DF38            next char

DF49  AF         xor a              reset flag
DF4A  32 39 AE   ld (AE39),a        flag used assembling a basic line
DF4D  C9         ret

      @ DF09!
DF4E  C5         push bc
DF4F  D5          push de
DF50  E5           push hl
DF51  CD 16 AC      call AC16       Indirection: Get a token while assembling
DF54  7E            ld a,(hl)       get first letter
DF55  23            inc hl
DF56  CD 8A FF      call FF8A       change <a> to upper case
DF59  CD DD E2      call E2DD       calculate TOKEN TABLE offset <de>
DF5C  CD 27 E3      call E327       find text(hl) within table(de)
DF5F  30 28         jr nc,DF89      not found
DF61  79            ld a,c
DF62  E6 7F         and 7F          mask out bit 7
DF64  CD 7B FF      call FF7B       test <a> for A-Z, =carry
DF67  30 0B         jr nc,DF74      not a letter
DF69  1A            ld a,(de)
DF6A  FE E4         cp E4           [FN]
DF6C  28 06         jr z,DF74       yes, FN
DF6E  7E            ld a,(hl)
DF6F  CD 7B FF      call FF7B       test <a> for A-Z, =carry
DF72  38 15         jr c,DF89       it is a letter
DF74  F1           pop af
DF75  1A            ld a,(de)
DF76  B7            or a
DF77  FA C8 DF      jp m,DFC8       ...
DF7A  D1          pop de
DF7B  C1         pop bc
DF7C  F5         push af
DF7D  3E FF       ld a,FF           [TOKEN SWITCH]
DF7F  CD 25 DF    call DF25         PUT CHAR <a> into Basic line (de), inc de, d
DF82  F1         pop af             insert TOKEN now
DF83  CD 25 DF   call DF25          PUT CHAR <a> into Basic line (de), inc de, d
DF86  AF         xor a
DF87  18 3A      jr DFC3            reset assembler flag

DF89  E1           pop hl
DF8A  D1          pop de
DF8B  C1         pop bc
DF8C  E5         push hl
DF8D  2B          dec hl
DF8E  23          inc hl
DF8F  7E          ld a,(hl)
DF90  CD 7B FF    call FF7B         test <a> for A-Z, =carry
DF93  38 F9       jr c,DF8E         it is a letter, get next
DF95  CD EA DF    call DFEA         if " or # ret z; set <a> $=3, %=2, &=1
DF98  38 04       jr c,DF9E         ...
DF9A  3E 0D       ld a,0D           <real var>
```

```
DF9C  18 06        jr DFA4          insert MARK <a> and 2 zero bytes

      @ DF98'
DF9E  23           inc hl
DF9F  FE 05        cp 05            is it real?
DFA1  20 01        jr nz,DFA4       insert MARK <a> and 2 zero bytes
DFA3  3D           dec a            set MARK to 4 if real

----- insert MARK <a> and 2 zero bytes
      @ DF9C' DFA1'
DFA4  CD 25 DF     call DF25        PUT CHAR <a> into Basic line (de), inc de, d
DFA7  AF           xor a
DFA8  CD 25 DF     call DF25        PUT CHAR <a> into Basic line (de), inc de, d
DFAB  AF           xor a
DFAC  CD 25 DF     call DF25        PUT CHAR <a> into Basic line (de), inc de, d
DFAF  E3           ex (sp),hl
DFB0  7E           ld a,(hl)
DFB1  CD 7B FF     call FF7B        test <a> for A-Z, =carry
DFB4  30 07        jr nc,DFBD       mark end of variable name
DFB6  7E           ld a,(hl)
DFB7  CD 25 DF     call DF25        PUT CHAR <a> into Basic line (de), inc de, d
DFBA  23           inc hl
DFBB  18 F3        jr DFB0          get next letter

----- mark end of variable name
DFBD  CD DF E0     call E0DF        mark end of name, set bit 7
DFC0  E1           pop hl
DFC1  3E FF        ld a,FF          set flag
DFC3  32 39 AE     ld (AE39),a      flag used assembling a basic line
DFC6  37           scf
DFC7  C9           ret

      @ DF77
DFC8  E5             push hl
DFC9  4F               ld c,a       look for a direct command
DFCA  21 DC DF         ld hl,DFDC   token: RESTORE RENUM DELETE EDIT RESUME ...
DFCD  CD AA FF         call FFAA    search <a> in table(hl); =carry
DFD0  9F               sbc a,a
DFD1  E6 01            and 01
DFD3  32 39 AE         ld (AE39),a  flag used assembling a basic line
DFD6  79               ld a,c
DFD7  E1             pop hl
DFD8  D1            pop de
DFD9  C1           pop bc
DFDA  B7           or a
DFDB  C9           ret

----- token: RESTORE RENUM DELETE EDIT RESUME ...
      @ DFCA:
DFDC  C7           [RESTORE]
DFDD  81           [AUTO]
DFDE  C6           [RENUM]
DFDF  92           [DELETE]
DFE0  96           [EDIT]
DFE1  C8           [RESUME]
DFE2  E3           [ERL]
DFE3  97           [ELSE]
DFE4  CA           [RUN]
DFE5  A7           [LIST]
DFE6  A0           [GOTO]
DFE7  EB           [THEN]
DFE8  9F           [GOSUB]
DFE9  00
```

```
----- if " or # ret z; set <a> $=3, %=2, &=1
      @ DF95! E1B4!
DFEA  FE 26       cp 26            '&
DFEC  D0          ret nc
DFED  FE 21       cp 21            '!
DFEF  3F          ccf
DFF0  D0          ret nc
DFF1  FE 22       cp 22            '"
DFF3  C8          ret z
DFF4  FE 23       cp 23            '#
DFF6  C8          ret z
DFF7  EE 27       xor 27           24=3, 25=2, 26=1
DFF9  FE 04       cp 04            sets carry
DFFB  CE FF       adc a,FF         leaves a as it was after xor 27
DFFD  37          scf
DFFE  C9          ret

----- it is a numeric value
      @ DEEF
DFFF  3A 39 AE    ld a,(AE39)      flag used assembling a basic line
E002  B7          or a
E003  28 15       jr z,E01A
E005  7E          ld a,(hl)
E006  23          inc hl
E007  FA 25 DF    jp m,DF25        PUT CHAR <a> into Basic line (de), inc de, d
E00A  FE 2E       cp 2E            '.
E00C  CA 25 DF    jp z,DF25        PUT CHAR <a> into Basic line (de), inc de, d
E00F  2B          dec hl
E010  D5          push de          pointer to Basic line
E011  CD 04 EE     call EE04       ...
E014  30 34        jr nc,E04A      copy edit buffer to Basic line till <hl>=<de
E016  3E 1E        ld a,1E         <next LINE#>
E018  18 4F        jr E069         copy <a> and <VARTPE> bytes from FAC to prog

E01A  D5          push de
E01B  C5           push bc
E01C  CD BE EC      call ECBE      get either HEX or integer VAL
E01F  C1           pop bc
E020  30 28        jr nc,E04A      copy edit buffer to Basic line till <hl>=<de
E022  CD 27 FF     call FF27       get VARTYPE <a>; cp string
E025  3E 1F        ld a,1F         <next 5 byte REAL>
E027  30 40        jr nc,E069      = real
E029  EB           ex de,hl
E02A  2A C2 B0     ld hl,(B0C2)    Floating point ACU, FAC
E02D  EB           ex de,hl
E02E  7A           ld a,d
E02F  B7           or a
E030  3E 1A        ld a,1A         <next 2 byte VAL>
E032  20 35        jr nz,E069      copy <a> and <VARTPE> bytes from FAC to prog
E034  E3           ex (sp),hl
E035  EB           ex de,hl
E036  7D           ld a,l
E037  FE 0A        cp 0A           >10.?
E039  30 04        jr nc,E03F      ...
E03B  C6 0E        add a,0E
E03D  18 06        jr E045         ...

E03F  3E 19        ld a,19         <next byte VAL>
E041  CD 25 DF     call DF25       PUT CHAR <a> into Basic line (de), inc de, d
E044  7D           ld a,l
E045  CD 25 DF     call DF25       PUT CHAR <a> into Basic line (de), inc de, d
E048  E1          pop hl
E049  C9          ret
```

```
----- copy edit buffer to Basic line till <hl>=<de>
      @ E014' E020' E056' E060'
E04A  7E          ld a,(hl)       get char from edit buffer
E04B  23          inc hl
E04C  E3          ex (sp),hl      get pointer to Basic line
E04D  EB          ex de,hl
E04E  CD 25 DF    call DF25       PUT CHAR <a> into Basic line (de), inc de, d
E051  EB          ex de,hl
E052  E3          ex (sp),hl
E053  CD B8 FF    call FFB8       test HL=DE? (try hl-de)
E056  20 F2       jr nz,E04A      copy edit buffer to Basic line till <hl>=<de
E058  D1         pop de
E059  C9         ret

----- it is a &HEX value
      @ DEF4
E05A  D5         push de
E05B  C5          push bc
E05C  CD BE EC     call ECBE      get either HEX or integer VAL
E05F  C1          pop bc
E060  30 E8       jr nc,E04A      copy edit buffer to Basic line till <hl>=<de
E062  FE 02       cp 02           <integer VAR%>
E064  3E 1B       ld a,1B         <next 2 byte BIN>
E066  28 01       jr z,E069       copy <a> and <VARTPE> bytes from FAC to prog
E068  3C          inc a

----- copy <a> and <VARTPE> bytes from FAC to program line
      @ E018' E027' E032' E066'
E069  D1         pop de
E06A  CD 25 DF   call DF25        PUT CHAR <a> into Basic line (de), inc de, d
E06D  E5         push hl
E06E  21 C2 B0    ld hl,B0C2      Floating point ACU, FAC
E071  CD 23 FF    call FF23       get VARTYPE <a>;
E074  F5          push af
E075  7E           ld a,(hl)
E076  23           inc hl
E077  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E07A  F1          pop af
E07B  3D          dec a
E07C  20 F6       jr nz,E074      insert next byte
E07E  E1         pop hl
E07F  C9         ret

----- it's neither letter nor digit nor blank
      @ DEFD
E080  FE 22      cp 22            '"
E082  28 3B      jr z,E0BF        copy text up to '" or end of buffer
E084  FE 7C      cp 7C            '|
E086  28 45      jr z,E0CD        it's an External Command
E088  C5         push bc
E089  D5          push de
E08A  EE 3F        xor 3F         '?
E08C  06 BF        ld b,BF        [PRINT]
E08E  28 16        jr z,E0A6      insert token for PRINT
E090  2B           dec hl
E091  11 4B E6     ld de,E64B     table of operators ^,\,>=,=>,>,=,<>,<=,=<,<,
E094  CD 27 E3     call E327      find text(hl) within table(de)
E097  1A           ld a,(de)
E098  38 08        jr c,E0A2      ...
E09A  7E           ld a,(hl)
E09B  FE 20        cp 20          'SPACE
E09D  30 02        jr nc,E0A1     ...
E09F  3E 20        ld a,20        'SPACE
E0A1  23           inc hl
E0A2  47           ld b,a
```

```
E0A3  CD B3 E0     call E0B3      test end of text or buffer
E0A6  32 39 AE     ld (AE39),a    flag used assembling a basic line
E0A9  78           ld a,b
E0AA  D1           pop de
E0AB  C1           pop bc
E0AC  FE C0        cp C0          [REM']
E0AE  28 36        jr z,E0E6      insert 01, 'REM and text following
E0B0  C3 25 DF     jp DF25        PUT CHAR <a> into Basic line (de), inc de, d

----- test end of text or buffer
      @ E0A3!
E0B3  3D           dec a
E0B4  C8           ret z
E0B5  EE 22        xor 22         '"
E0B7  C8           ret z
E0B8  3A 39 AE     ld a,(AE39)    flag used assembling a basic line
E0BB  3C           inc a
E0BC  C8           ret z
E0BD  3D           dec a
E0BE  C9           ret

----- copy text up to '" or end of buffer
      @ DF44! E082' E0C8'
E0BF  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E0C2  7E           ld a,(hl)
E0C3  B7           or a
E0C4  C8           ret z          end of buffer
E0C5  23           inc hl
E0C6  FE 22        cp 22          '"
E0C8  20 F5        jr nz,E0BF     copy text up to '" or end of buffer
E0CA  C3 25 DF     jp DF25        PUT CHAR <a> into Basic line (de), inc de, d

----- it's an External Command
      @ E086'
E0CD  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E0D0  AF           xor a          reset
E0D1  32 39 AE     ld (AE39),a    flag used assembling a basic line
E0D4  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E0D7  7E           ld a,(hl)
E0D8  23           inc hl
E0D9  CD 7B FF     call FF7B      test <a> for A-Z, =carry
E0DC  38 F6        jr c,E0D4      skip over NAME
E0DE  2B           dec hl

----- mark end of name, set bit 7
      @ DFBD!
E0DF  1B           dec de         get last char
E0E0  1A           ld a,(de)
E0E1  F6 80        or 80          set bit 7 to mark end
E0E3  12           ld (de),a      and store back
E0E4  13           inc de
E0E5  C9           ret

----- insert 01, 'REM and text following
E0E6  3E 01        ld a,01        <statement end>
E0E8  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E0EB  3E C0        ld a,C0        [REM']

----- copy REMark to basic text
      @ DF0F  E0F3'
E0ED  CD 25 DF     call DF25      PUT CHAR <a> into Basic line (de), inc de, d
E0F0  7E           ld a,(hl)
E0F1  23           inc hl
E0F2  B7           or a
E0F3  20 F8        jr nz,E0ED     copy REMark to basic text
```

```
E0F5  2B          dec hl
E0F6  C9          ret

----- command: LIST [<line>[-<line>]][,#<device>]
      @ DE4F
E0F7  CD B0 CE    call CEB0         get line#'s, default <bc>=1, <de>=65535.
E0FA  C5          push bc
E0FB  D5           push de
E0FC  CD C6 C1      call C1C6       if '# get chan; default=0; set output channe
E0FF  CD 4A DD      call DD4A       CHRGOT <a>; end of statement? else syntax er
E102  CD CB DD      call DDCB       reset BASIC program counter
E105  D1           pop de
E106  C1          pop bc
E107  CD 0D E1    call E10D         perform LIST
E10A  C3 64 C0    jp C064           reset Basic

----- perform LIST
      @ E107! EC97!
E10D  D5          push de
E10E  50           ld d,b
E10F  59           ld e,c           de=bc = start line#
E110  CD A3 E7     call E7A3        search line# <de> from start, <hl>=address,
E113  D1          pop de
E114  4E          ld c,(hl)
E115  23          inc hl
E116  46          ld b,(hl)         bc= len of this line
E117  2B          dec hl
E118  78          ld a,b
E119  B1          or c
E11A  C8          ret z             end of program
E11B  CD 3C C4    call C43C         check for a BREAK request
E11E  E5          push hl
E11F  09           add hl,bc        add len of line
E120  E3           ex (sp),hl
E121  D5           push de
E122  E5            push hl
E123  23             inc hl
E124  23             inc hl
E125  5E             ld e,(hl)      <de>= line#
E126  23             inc hl
E127  56             ld d,(hl)
E128  E1            pop hl
E129  E3            ex (sp),hl
E12A  CD B8 FF      call FFB8       test HL=DE? (try hl-de)
E12D  E3            ex (sp),hl
E12E  38 12         jr c,E142       last line# done
E130  CD 63 E1      call E163       list a basic line into the edit buffer
E133  CD 45 E1      call E145       print a char (hl) to output channel
E136  23            inc hl
E137  7E            ld a,(hl)
E138  B7            or a
E139  20 F8         jr nz,E133      print next char
E13B  CD 4E C3      call C34E       output 'LF to channel
E13E  D1           pop de
E13F  E1          pop hl
E140  18 D2       jr E114           do the next line

E142  E1           pop hl
E143  E1          pop hl
E144  C9          ret
```

```
----- print a char (hl) to output channel
      @ E133!
E145  CD BA C1      call C1BA         get output channel; cp 08
E148  38 0B         jr c,E155         print to screen
E14A  7E            ld a,(hl)
E14B  CD 6E C3      call C36E         output char <a> to channel
E14E  FE 0A         cp 0A             'LF  (^J)
E150  C0            ret nz
E151  3E 0D         ld a,0D           'CR  (^M)
E153  18 0B         jr E160           jp output char <a> to channel

E155  7E            ld a,(hl)
E156  FE 20         cp 20             'SPACE
E158  30 06         jr nc,E160        jp output char <a> to channel
E15A  3E 01         ld a,01           'SOH (^A)
E15C  CD 6E C3      call C36E         output char <a> to channel
E15F  7E            ld a,(hl)

----- jp output char <a> to channel
E160  C3 6E C3      jp C36E           output char <a> to channel

----- list a basic line into the edit buffer
      @ C05C! E130!
E163  D5            push de
E164  01 A4 AC       ld bc,ACA4       EDIT BUFFER
E167  C5              push bc
E168  23               inc hl
E169  23               inc hl
E16A  5E               ld e,(hl)
E16B  23               inc hl
E16C  56               ld d,(hl)      de = line#
E16D  23               inc hl
E16E  E5               push hl
E16F  EB                 ex de,hl
E170  CD 0D FF           call FF0D    set FAC to <hl> and mark integer
E173  CD 82 EE           call EE82    convert FAC to ASCII, (hl)=address of text
E176  11 00 00           ld de,0000
E179  7E                 ld a,(hl)    copy line# to edit buffer
E17A  23                 inc hl
E17B  B7                 or a
E17C  28 05              jr z,E183    end of line#
E17E  CD FE E1           call E1FE    PUTCHAR to (bc), update count and pointer
E181  18 F6              jr E179      next

E183  3E 20              ld a,20      insert a space after line#
E185  CD FE E1           call E1FE    PUTCHAR to (bc), update count and pointer
E188  E1               pop hl
E189  7E               ld a,(hl)
E18A  B7               or a
E18B  28 05            jr z,E192      end of line
E18D  CD 96 E1         call E196      expand basic code, copy to edit buffer
E190  18 F7            jr E189        next

E192  02               ld (bc),a
E193  E1              pop hl
E194  D1            pop de
E195  C9            ret

----- expand basic code, copy to edit buffer
      @ E18D!
E196  CD 13 AC      call AC13         Indirection: LIST and EDIT
E199  FA 20 E2      jp m,E220         expand a token to its TEXT
E19C  FE 02         cp 02             <integer VAR%>
E19E  38 1D         jr c,E1BD         it is a statement separator ':
E1A0  FE 05         cp 05             <FAC real>
```

```
E1A2  38 43      jr c,E1E7       expand a VARIABLE
E1A4  FE 0B      cp 0B           <integer var>
E1A6  38 22      jr c,E1CA       copy char <a> or "text"
E1A8  FE 0E      cp 0E           <const 0>
E1AA  38 3B      jr c,E1E7       expand a VARIABLE
E1AC  FE 20      cp 20           'SPACE
E1AE  38 2E      jr c,E1DE       expand a CONSTANT value
E1B0  FE 7C      cp 7C           '|
E1B2  28 51      jr z,E205       insert EXTERNAL COMMAND introducer '|
E1B4  CD EA DF   call DFEA       if " or # ret z; set <a> $=3, %=2, &=1
E1B7  DC 1A E2   call c,E21A     insert a space if <e>=1 and return
E1BA  7E         ld a,(hl)
E1BB  18 0D      jr E1CA         copy char <a> or "text"

----- it is a statement separator ':
E1BD  23         inc hl
E1BE  7E         ld a,(hl)
E1BF  FE C0      cp C0           [REM']
E1C1  28 5D      jr z,E220       expand a token to its TEXT
E1C3  FE 97      cp 97           [ELSE]
E1C5  28 59      jr z,E220       expand a token to its TEXT
E1C7  2B         dec hl
E1C8  3E 3A      ld a,3A         ':

----- copy char <a> or "text"
E1CA  1E 00      ld e,00         flag; no space necessary
E1CC  FE 22      cp 22           '"
E1CE  20 0B      jr nz,E1DB      insert char and return
E1D0  CD FE E1   call E1FE       PUTCHAR to (bc), update count and pointer
E1D3  23         inc hl
E1D4  7E         ld a,(hl)       copy text between <"> as it is
E1D5  B7         or a
E1D6  C8         ret z
E1D7  FE 22      cp 22           '"
E1D9  20 F5      jr nz,E1D0      next text char
E1DB  23         inc hl
E1DC  18 20      jr E1FE         PUTCHAR to (bc), update count and pointer

----- expand a CONSTANT value
E1DE  CD 1A E2   call E21A       insert a space if <e>=1 and return
E1E1  CD 53 E2   call E253       convert CONSTANT to ascii, according MARK
E1E4  1E 01      ld e,01         flag; append a space
E1E6  C9         ret

----- expand a VARIABLE
      @ E1A2' E1AA'
E1E7  CD 1A E2   call E21A       insert a space if <e>=1 and return
E1EA  7E         ld a,(hl)
E1EB  F5         push af
E1EC  23          inc hl
E1ED  23          inc hl
E1EE  23          inc hl
E1EF  CD 0F E2    call E20F      remove bit 7 from last char
E1F2  F1         pop af
E1F3  1E 01      ld e,01         flag; append space
E1F5  FE 0B      cp 0B           <integer var>
E1F7  D0         ret nc
E1F8  1E 00      ld e,00         flag: no space
E1FA  EE 27      xor 27          produce % $ !
E1FC  E6 FD      and FD          enable upper ROM, disable lower ROM
```

```
----- PUTCHAR to (bc), update count and pointer
      @ E17E! E185! E1D0! E1DC' E207! E212! E21E' E234! E23C! E2BD! E2C3!
      @ E2D4!
E1FE  02          ld (bc),a
E1FF  03          inc bc
E200  15          dec d
E201  C0          ret nz
E202  0B          dec bc
E203  14          inc d
E204  C9          ret

----- insert EXTERNAL COMMAND introducer '|
E205  1E 01       ld e,01           Unexpected NEXT
E207  CD FE E1    call E1FE         PUTCHAR to (bc), update count and pointer
E20A  23          inc hl
E20B  7E          ld a,(hl)
E20C  23          inc hl
E20D  B7          or a
E20E  C0          ret nz

----- remove bit 7 from last char
      @ E1EF! E217'
E20F  7E          ld a,(hl)
E210  E6 7F       and 7F            mask out
E212  CD FE E1    call E1FE         PUTCHAR to (bc), update count and pointer
E215  BE          cp (hl)
E216  23          inc hl
E217  30 F6       jr nc,E20F        remove bit 7 from last char
E219  C9          ret

----- insert a space if <e>=1 and return
      @ E1B7! E1DE! E1E7! E230!
E21A  1D          dec e
E21B  C0          ret nz
E21C  3E 20       ld a,20           'SPACE
E21E  18 DE       jr E1FE           PUTCHAR to (bc), update count and pointer

----- expand a token to its TEXT
      @ E199  E1C1' E1C5'
E220  23          inc hl
E221  FE FF       cp FF             [TOKEN SWITCH]
E223  20 02       jr nz,E227        insert first letter of TOKEN-NAME
E225  7E          ld a,(hl)
E226  23          inc hl            skip over token SWITCH
E227  F5          push af
E228  E5           push hl
E229  CD ED E2      call E2ED       search TOKEN or OPERATOR <a> in table
E22C  B7            or a
E22D  28 08         jr z,E237       token not found
E22F  F5            push af
E230  CD 1A E2       call E21A      insert a space if <e>=1 and return
E233  F1            pop af
E234  CD FE E1      call E1FE       PUTCHAR to (bc), update count and pointer
E237  7E            ld a,(hl)
E238  E6 7F         and 7F          mask out bit 7
E23A  FE 09         cp 09           skip over 09 (allowing blanks on compare)
E23C  C4 FE E1      call nz,E1FE    PUTCHAR to (bc), update count and pointer
E23F  BE            cp (hl)
E240  23            inc hl
E241  28 F4         jr z,E237       next NAME char
E243  CD 7B FF      call FF7B       test <a> for A-Z, =carry
E246  1E 00         ld e,00         flag: append no space
E248  30 02         jr nc,E24C
E24A  1E 01         ld e,01         flag: append space
E24C  E1           pop hl
```

```
E24D  F1          pop af
E24E  D6 E4       sub E4             [FN]
E250  C0          ret nz
E251  5F          ld e,a             flag: 0 = append no space after FN
E252  C9          ret

----- convert CONSTANT to ascii, according MARK
      @ E1E1!
E253  D5          push de
E254  7E           ld a,(hl)
E255  23           inc hl
E256  FE 1B        cp 1B             <next 2 byte BIN>
E258  28 49        jr z,E2A3         edit in binary representation
E25A  FE 1C        cp 1C             <next 2 byte HEX>
E25C  28 50        jr z,E2AE         edit in hex representation
E25E  FE 1E        cp 1E             <next LINE#>
E260  28 26        jr z,E288         edit a line#
E262  FE 1D        cp 1D             <next ADDRESS>
E264  28 22        jr z,E288         edit a line#
E266  FE 1F        cp 1F             <next 5 byte REAL>
E268  28 5E        jr z,E2C8         edit a real constant
E26A  FE 19        cp 19             <next byte VAL>
E26C  28 09        jr z,E277         edit a one byte value
E26E  FE 1A        cp 1A             <next 2 byte VAL>
E270  28 0B        jr z,E27D         edit a signed integer constant

----- VAL = MARK-0E, range 0-9
E272  D6 0E       sub 0E
E274  5F          ld e,a
E275  18 02       jr E279

----- edit a one byte value
E277  5E           ld e,(hl)
E278  23           inc hl
E279  16 00        ld d,00
E27B  18 04        jr E281

----- edit a signed integer constant
E27D  5E           ld e,(hl)
E27E  23           inc hl
E27F  56           ld d,(hl)
E280  23           inc hl
E281  E3           ex (sp),hl
E282  EB           ex de,hl
E283  CD 0D FF     call FF0D         set FAC to <hl> and mark integer
E286  18 47        jr E2CF           ...

----- edit a line#
      @ E260' E264'
E288  5E           ld e,(hl)
E289  23           inc hl
E28A  56           ld d,(hl)
E28B  23           inc hl
E28C  FE 1E        cp 1E             <next LINE#>
E28E  28 09        jr z,E299

----- get line# from line address
E290  E5          push hl            <de> is the line#
E291  EB           ex de,hl
E292  23           inc hl
E293  23           inc hl
E294  23           inc hl
E295  5E           ld e,(hl)
E296  23           inc hl
E297  56           ld d,(hl)
```

```
E298  E1          pop hl
E299  E3          ex (sp),hl
E29A  EB          ex de,hl
E29B  CD 0D FF    call FF0D         set FAC to <hl> and mark integer
E29E  CD 82 EE    call EE82         convert FAC to ASCII, (hl)=address of text
E2A1  18 2F       jr E2D2           copy NUMBER EDIT BUFFER to EDIT BUFFER

----- edit in binary representation
E2A3  C5           push bc
E2A4  01 02 00      ld bc,0002      b= min # of digits, c=vartype of source
E2A7  CD 14 F1      call F114       edit value in binary representation
E2AA  3E 58         ld a,58         'X
E2AC  18 09         jr E2B7

----- edit in hex representation
E2AE  C5           push bc
E2AF  01 02 00      ld bc,0002      b= min # of digits, c=vartype of source
E2B2  CD 19 F1      call F119       edit value in HEX representation
E2B5  3E 48         ld a,48         'H
E2B7  C1           pop bc
E2B8  E3           ex (sp),hl       de points to NUMBER EDIT BUFFER
E2B9  EB           ex de,hl
E2BA  F5           push af
E2BB  3E 26         ld a,26         '&
E2BD  CD FE E1      call E1FE       PUTCHAR to (bc), update count and pointer
E2C0  F1           pop af
E2C1  FE 48        cp 48            'H
E2C3  C4 FE E1     call nz,E1FE     PUTCHAR to (bc), update count and pointer
E2C6  18 0A        jr E2D2          copy NUMBER EDIT BUFFER to EDIT BUFFER

----- edit a real constant
E2C8  3E 05        ld a,05          <FAC real>
E2CA  CD 4B FF     call FF4B        set VARTYPE <a>, copy VARIABLE (hl) to FAC
E2CD  E3           ex (sp),hl
E2CE  EB           ex de,hl

      @ E286'
E2CF  CD 8F EE     call EE8F        edit FAC onto the NUMBER EDIT BUFFER

----- copy NUMBER EDIT BUFFER to EDIT BUFFER
E2D2  7E           ld a,(hl)
E2D3  23           inc hl
E2D4  CD FE E1     call E1FE        PUTCHAR to (bc), update count and pointer
E2D7  7E           ld a,(hl)
E2D8  B7           or a
E2D9  20 F7         jr nz,E2D2      copy NUMBER EDIT BUFFER to EDIT BUFFER
E2DB  E1          pop hl
E2DC  C9          ret

----- calculate TOKEN TABLE offset <de>
      @ DF59!
E2DD  E5          push hl
E2DE  D6 41        sub 41           remove ascii part
E2E0  87           add a,a          *2
E2E1  C6 54        add a,54         + lowbyte of address
E2E3  6F           ld l,a
E2E4  CE E3        adc a,E3         highbyte of address
E2E6  95           sub l
E2E7  67           ld h,a           highbyte including a possible carry
E2E8  5E           ld e,(hl)        get start address of letter
E2E9  23           inc hl
E2EA  56           ld d,(hl)
E2EB  E1          pop hl
E2EC  C9          ret
```

```
----- search TOKEN or OPERATOR <a> in table
      @ E229!
E2ED  C5          push bc
E2EE  4F           ld c,a          c = token to search
E2EF  06 1A        ld b,1A         =26.
E2F1  21 88 E3     ld hl,E388      'Token Tabelle
E2F4  CD 13 E3     call E313       get pointer to TOKEN TEXT, =carry
E2F7  38 0D        jr c,E306       token found
E2F9  23           inc hl
E2FA  10 F8        djnz E2F4       check next
E2FC  21 4B E6     ld hl,E64B      table of operators ^,\,>=,=>,>,=,<>,<=,=<,<,
E2FF  CD 13 E3     call E313       get pointer to TOKEN TEXT, =carry
E302  30 07        jr nc,E30B
E304  06 C0        ld b,C0         c0+40=100, a=0, carry =set
E306  78           ld a,b
E307  C6 40        add a,40        add on ascii part for first letter
E309  C1          pop bc
E30A  C9          ret

E30B  CD 19 AC     call AC19       Indirection: Token not found on LIST
E30E  1E 02        ld e,02         Syntax error
E310  C3 94 CA     jp CA94         perform ERROR <e> routine

----- get pointer to TOKEN TEXT, =carry
      @ E2F4! E2FF! E322'
E313  7E          ld a,(hl)
E314  B7          or a             table char
E315  C8          ret z            end of table
E316  E5          push hl
E317  7E           ld a,(hl)       table char
E318  23           inc hl
E319  17           rla
E31A  30 FB        jr nc,E317      skip over this entry
E31C  7E           ld a,(hl)       this is the token after entry
E31D  23           inc hl
E31E  B9           cp c            is it the wanted one?
E31F  28 03        jr z,E324       yes, return with carry set
E321  F1          pop af
E322  18 EF       jr E313          get pointer to TOKEN TEXT, =carry

E324  E1          pop hl
E325  37          scf
E326  C9          ret

----- find text(hl) within table(de)
      @ DF5C! E094! E34F'
E327  1A          ld a,(de)        table char
E328  B7          or a
E329  C8          ret z            end of table
E32A  E5          push hl
E32B  1A           ld a,(de)       table char
E32C  13           inc de
E32D  FE 09        cp 09           this code allows for spaces
E32F  28 04        jr z,E335       skip over blanks in text
E331  FE 20        cp 20           'SPACE
E333  20 05        jr nz,E33A      no blank
E335  CD 61 DD     call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
E338  18 F1        jr E32B         next char

E33A  4F           ld c,a          save table char
E33B  7E           ld a,(hl)       get text char
E33C  23           inc hl
E33D  CD 8A FF     call FF8A       change <a> to upper case
E340  A9           xor c           are they the same?
E341  28 E8        jr z,E32B       compare next
```

```
E343  E6 7F       and 7F          mask out bit 7 (mark for last char)
E345  28 0A       jr z,E351       now it's a match
E347  1B          dec de
E348  1A          ld a,(de)       get last but one
E349  13          inc de
E34A  17          rla             test for bit 7 in table entry
E34B  30 FB       jr nc,E348      skip over this entry
E34D  13          inc de
E34E  E1         pop hl
E34F  18 D6      jr E327          find text(hl) within table(de)

E351  F1         pop af
E352  37         scf
E353  C9         ret

----- addresses of first letter
      @ hidden reference DF59:
E354  35 E6      E635             'A
E356  2A E6      E62A             'B
E358  EF E5      E5EF             'C
E35A  B9 E5      E5B9             'D
E35C  8A E5      E58A             'E
E35E  7E E5      E57E             'F
E360  72 E5      E572             'G
E362  68 E5      E568             'H
E364  47 E5      E547             'I
E366  43 E5      E543             'J
E368  3F E5      E53F             'K
E36A  13 E5      E513             'L
E36C  ED E4      E4ED             'M
E36E  E2 E4      E4E2             'N
E370  AA E4      E4AA             'O
E372  86 E4      E486             'P
E374  85 E4      E485             'Q
E376  3B E4      E43B             'R
E378  FB E3      E3FB             'S
E37A  CF E3      E3CF             'T
E37C  C0 E3      E3C0             'U
E37E  B8 E3      E3B8             'V
E380  9A E3      E39A             'W
E382  92 E3      E392             'X
E384  8D E3      E38D             'Y
E386  88 E3      E388             'Z
```

```
----- 'Token Tabelle
      @ E2F1:
E388  4F 4E C5 DA 00                                     'ONEZ.
E38D  50 4F D3 48 00                                     'POSH.
E392  50 4F D3 47 4F D2 FD 00                            'POSGOR}.
E39A  52 49 54 C5 D9 49 4E 44  4F D7 D8 49 44 54 C8 D7   'RITEYINDOWXIDTHW
E3AA  48 49 4C C5 D6 45 4E C4  D5 41 49 D4 D4 00         'HILEVENDUAITT.
E3B8  50 4F D3 7F 41 CC 1D 00                            'POS.AL..
E3C0  53 49 4E C7 ED 50 50 45  52 A4 1C 4E D4 1B 00      'SINGmPPER$.NT..
E3CF  52 4F CE D3 52 4F 46 C6  D2 CF EC 49 4D C5 46 48   'RONSROFFROlIMEFH
E3DF  45 CE EB 45 53 54 D2 7D  45 53 D4 7C 41 CE 1A 41   'ENkESTR}EST|AN.A
E3EF  47 4F 46 C6 D1 41 C7 D0  41 C2 EA 00               'GOFFQAGPABj.
E3FB  59 4D 42 4F CC CF 57 41  D0 E7 54 52 49 4E 47 A4   'YMBOLOWAPgTRING$
E40B  7B 54 52 A4 19 54 4F D0  CE 54 45 D0 E6 51 D2 18   '{TR$.TOPNTEPfQR.
E41B  D1 17 50 45 45 C4 CD 50  C3 E5 50 41 43 45 A4 16   'Q.PEEDMPCePACE$.
E42B  4F 55 4E C4 CC 49 CE 15  47 CE 14 41 56 C5 CB 00   'OUNDLIN.GN.AVEK.
E43B  55 CE CA 4F 55 4E C4 7A  4E C4 45 49 47 48 54 A4   'UNJOUNDzNDEIGHT$
E44B  79 45 54 55 52 CE C9 45  53 55 4D C5 C8 45 53 54   'yETURNIESUMEHEST
E45B  4F 52 C5 C7 45 4E 55 CD  C6 45 4D 41 49 CE 13 45   'OREGENUMFEMAIN.E
E46B  CD C5 45 4C 45 41 53 C5  C4 45 41 C4 C3 41 4E 44   'MEELEASEDEADCAND
E47B  4F 4D 49 5A C5 C2 41 C4  C1 00                     'OMIZEBADA.
E485  00                                                 '.
E486  52 49 4E D4 BF 4F D3 78  4F 4B C5 BE 4C 4F 54 D2   'RINT?OSxOKE>LOTR
E496  BD 4C 4F D4 BC C9 44 45  CE BB 45 45 CB 12 41 50   '=LOT<IDEN;EEK.AP
E4A6  45 D2 BA 00                                        'ER:.
E4AA  55 D4 B9 52 49 47 49 CE  B8 D2 FC 50 45 4E 4F 55   'UT9RIGIN8R|PENOU
E4BA  D4 B7 50 45 4E 49 CE B6  4E 20 53 D1 B5 4E 20 45   'T7PENIN6N SQ5N E
E4CA  52 52 4F 52 20 47 4F 09  54 4F 20 B0 B4 4E 20 42   'RROR GO.TO 04N B
E4DA  52 45 41 CB B3 CE B2 00                            'REAK3N2.
E4E2  4F D4 FE 45 D7 B1 45 58  D4 B0 00                  'OT~EW1EXT0.
E4ED  4F 56 45 D2 AF 4F 56 C5  AE 4F 44 C5 AD 4F C4 FB   'OVER/OVE.ODE-OD{
E4FD  49 CE 77 49 44 A4 AC 45  52 47 C5 AB 45 4D 4F 52   'INwID$,ERGE+EMOR
E50D  D9 AA 41 D8 76 00                                  'Y*AXv.
E513  4F 57 45 52 A4 11 4F 47  31 B0 10 4F C7 0F 4F 43   'OWER$.OG10.OG.OC
E523  41 54 C5 A9 4F 41 C4 A8  49 53 D4 A7 49 4E C5 A6   'ATE)OAD(IST'INE&
E533  45 D4 A5 45 CE 0E 45 46  54 A4 75 00               'ET%EN.EFT$u.
E53F  45 D9 A4 00                                        'EY$.
E543  4F D9 0D 00                                        'OY..
E547  4E D4 0C 4E 53 54 D2 74  4E 50 55 D4 A3 4E D0 0B   'NT.NSTRtNPUT#NP.
E557  4E 4B 45 59 A4 43 4E 4B  45 D9 0A 4E CB A2 C6 A1   'NKEY$CNKEY.NK"F!
E567  00                                                 '.
E568  49 4D 45 CD 42 45 58 A4  73 00                     'IMEMBEX$s.
E572  4F 09 54 CF A0 4F 09 53  55 C2 9F 00               'O.TO O.SUB..
E57E  52 C5 09 4F D2 9E CE E4  49 D8 08 00               'RE.OR.NdIX..
E58A  58 D0 07 56 45 52 D9 9D  52 52 4F D2 9C 52 D2 41   'XP.VERY.RROR.RRA
E59A  52 CC E3 52 41 53 C5 9B  4F C6 40 4E D6 9A 4E D4   'RLcRASE.OF@NV.NT
E5AA  99 4E C4 98 4C 53 C5 97  C9 DC 44 49 D4 96 00      '.ND.LSE.I\DIT..
E5B9  52 41 57 D2 95 52 41 D7  94 49 CD 93 C9 DB 45 4C   'RAWR.RAW.IM.I[EL
E5C9  45 54 C5 92 45 C7 91 45  46 53 54 D2 90 45 46 52   'ETE.EG.EFSTR.EFR
E5D9  45 41 CC 8F 45 46 49 4E  D4 8E 45 C6 8D 45 43 A4   'EAL.EFINT.EF.EC$
E5E9  72 41 54 C1 8C 00                                  'rATA..
E5EF  52 45 41 CC 06 4F D3 05  4F 4E D4 8B 4C D3 8A 4C   'REAL.OS.ONT.LS.L
E5FF  4F 53 45 4F 55 D4 89 4C  4F 53 45 49 CE 88 4C C7   'OSEOUT.LOSEIN.LG
E60F  87 4C 45 41 D2 86 49 4E  D4 04 48 52 A4 03 48 41   '.LEAR.INT.HR$.HA
E61F  49 CE 85 41 D4 84 41 4C  CC 83 00                  'IN.AT.ALL..
E62A  4F 52 44 45 D2 82 49 4E  A4 71 00                  'ORDER.IN$q.
E635  55 54 CF 81 54 CE 02 53  C3 01 4E C4 FA 46 54 45   'UTO.TN.SC.NDzFTE
E645  D2 80 42 D3 00                                     'R.BS.
E64A  00                                                 '.

----- table of operators ^,\,>=,=>,>,=,<>,<=,=<,<,/,,*,-,+,'
      @ E091: E2FC:
E64B  DE F8 DC F9 3E 09 BD F0  3D 20 BE F0 BE EE BD EF   '^x\y>.=p= >p>n=o
E65B  3C 09 BE F2 3C 09 BD F3  3D 20 BC F3 BC F1 AF F7   '<.>r<.=s= <s<q/w
E66B  BA 01 AA F6 AD F5 AB F4  A7 C0 00                  ':.*v-u+t'@.
```

```
----- reset line MARK, basic end=start
      @ C152! C174!
E676  AF          xor a             reset
E677  32 3A AE    ld (AE3A),a       BASIC Program line format
E67A  2A 81 AE    ld hl,(AE81)      start of BASIC program -1 pointer
E67D  77          ld (hl),a
E67E  23          inc hl
E67F  77          ld (hl),a
E680  23          inc hl
E681  77          ld (hl),a
E682  23          inc hl
E683  22 83 AE    ld (AE83),hl      end of BASIC program pointer
E686  C9          ret

----- change Basic program to line# format
      @ E6D2! E75A! EAB8! EC44!
E687  3A 3A AE    ld a,(AE3A)       BASIC Program line format
E68A  B7          or a
E68B  C8          ret z             already in line# format
E68C  C5          push bc
E68D  D5           push de
E68E  E5            push hl
E68F  01 9D E6        ld bc,E69D    change MARK <next addr> to <next line#>
E692  CD FF E8        call E8FF     go thru Basic program and do function <bc>
E695  AF              xor a         reset
E696  32 3A AE        ld (AE3A),a   BASIC Program line format
E699  E1            pop hl
E69A  D1           pop de
E69B  C1          pop bc
E69C  C9          ret

----- change MARK <next addr> to <next line#>
      @ E68F: E6A5' E6BA'
E69D  CD 43 E9    call E943         skip over a statement element
E6A0  FE 02       cp 02             end of statement?
E6A2  D8          ret c
E6A3  FE 1D       cp 1D             <next ADDRESS>
E6A5  20 F6       jr nz,E69D        it's not <next address> format, take next
E6A7  56          ld d,(hl)
E6A8  2B          dec hl
E6A9  5E          ld e,(hl)
E6AA  2B          dec hl
E6AB  E5          push hl
E6AC  EB           ex de,hl
E6AD  23           inc hl
E6AE  23           inc hl
E6AF  23           inc hl
E6B0  5E           ld e,(hl)
E6B1  23           inc hl
E6B2  56           ld d,(hl)
E6B3  E1          pop hl
E6B4  36 1E       ld (hl),1E        set MARK for line#, insert line#
E6B6  23          inc hl
E6B7  73          ld (hl),e
E6B8  23          inc hl
E6B9  72          ld (hl),d
E6BA  18 E1       jr E69D           change MARK <next addr> to <next line#>

----- check for a line# in command line
      @ C0B8! EBFB!
E6BC  CD 61 DD    call DD61         CHRSKIP <a>; skip over blank, tab, linefeed
E6BF  B7          or a
E6C0  37          scf
E6C1  C8          ret z
E6C2  CD 04 EE    call EE04         ...
```

```
E6C5  D0          ret nc
E6C6  7E          ld a,(hl)
E6C7  FE 20       cp 20            'SPACE
E6C9  20 01       jr nz,E6CC       ...
E6CB  23          inc hl

      @ E6C9'
E6CC  CD D2 E6    call E6D2        assemble an insert line into program
E6CF  37          scf
E6D0  9F          sbc a,a
E6D1  C9          ret

----- assemble andinsert line into program
      @ C0A8! E6CC!
E6D2  CD 87 E6    call E687        change Basic program to line# format
E6D5  CD BB DE    call DEBB        assemble a program line; (hl)=edit buffer
E6D8  E5          push hl
E6D9  CD 61 DD     call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
E6DC  B7           or a
E6DD  28 28        jr z,E707       ...
E6DF  C5           push bc
E6E0  D5            push de
E6E1  21 04 00       ld hl,0004    ...
E6E4  09             add hl,bc
E6E5  E5             push hl
E6E6  E5              push hl
E6E7  CD A3 E7         call E7A3   search line# <de> from start, <hl>=address,
E6EA  E5               push hl
E6EB  DC 0B E7          call c,E70Bdelete a line# from Basic program
E6EE  D1               pop de
E6EF  C1              pop bc
E6F0  CD F8 F5        call F5F8    allocate space for new variables
E6F3  CD 2C F5        call F52C    shift all Basic pointers up by <bc>
E6F6  EB              ex de,hl
E6F7  D1             pop de
E6F8  73             ld (hl),e
E6F9  23             inc hl
E6FA  72             ld (hl),d
E6FB  23             inc hl
E6FC  D1            pop de
E6FD  73            ld (hl),e
E6FE  23            inc hl
E6FF  72            ld (hl),d
E700  23            inc hl
E701  C1           pop bc
E702  EB           ex de,hl
E703  E1          pop hl
E704  C3 F2 FF    jp FFF2          ldir

      @ E6DD'
E707  E1          pop hl
E708  CD 9A E7    call E79A        search line# <de> from start, <hl>=addr, nc=

----- delete a line# from Basic program
      @ E6EB! E764
E70B  C5          push bc
E70C  E5           push hl
E70D  09            add hl,bc
E70E  EB            ex de,hl
E70F  2A 89 AE      ld hl,(AE89)   upper end of DIM'd variables pointer
E712  CD CF FF      call FFCF      HL=HL-DE
E715  44            ld b,h
E716  4D            ld c,l         bc=hl
E717  EB            ex de,hl
E718  D1           pop de
```

```
E719  78            ld a,b
E71A  B1            or c
E71B  C4 F2 FF      call nz,FFF2    ldir
E71E  D1           pop de
E71F  21 00 00     ld hl,0000
E722  CD DA FF     call FFDA        BC=HL-DE
E725  C3 2C F5     jp F52C          shift all Basic pointers up by <bc>

----- command: DELETE <line#>[-<line#>]
      @ DE25
E728  CD 37 E7     call E737        get bounds of the lines to delete
E72B  CD 4A DD     call DD4A        CHRGOT <a>; end of statement? else syntax er
E72E  CD 5A E7     call E75A        delete line area; shift rest of pgm down
E731  CD 7A C1     call C17A        reset pointers; program, error, data
E734  C3 64 C0     jp C064          reset Basic

----- get bounds of the lines to delete
      @ E728! EA65!
E737  CD B0 CE     call CEB0        get line#'s, default <bc>=1, <de>=65535.
E73A  E5           push hl
E73B  C5            push bc         save the lower line#
E73C  CD C1 E7       call E7C1      search line# <de>; <hl>=address
E73F  D1            pop de
E740  E5            push hl
E741  CD A3 E7       call E7A3      search line# <de> from start, <hl>=address,
E744  22 3B AE       ld (AE3B),hl   used by DELETE <line#>, lower addr
E747  EB             ex de,hl
E748  E1            pop hl
E749  CD CF FF      call FFCF       HL=HL-DE
E74C  22 3D AE      ld (AE3D),hl    used by DELETE <line#>, upper addr
E74F  38 04         jr c,E755       Error: Improper argument
E751  7C            ld a,h
E752  B5            or l
E753  E1           pop hl
E754  C0           ret nz

----- Error: Improper argument
E755  1E 05         ld e,05         Improper argument
E757  C3 94 CA      jp CA94         perform ERROR <e> routine

----- delete line area; shift rest of pgm down
      @ E72E! EA81!
E75A  CD 87 E6     call E687        change Basic program to line# format
E75D  ED 4B 3D AE ld bc,(AE3D)      used by DELETE <line#>, upper addr
E761  2A 3B AE     ld hl,(AE3B)     used by DELETE <line#>, lower addr
E764  C3 0B E7     jp E70B          delete a line# from Basic program

----- get address VAL into <de>
      @ C6E8! C6ED! C866! C8D6! CC09! E9E0!
E767  23           inc hl
E768  5E           ld e,(hl)
E769  23           inc hl
E76A  56           ld d,(hl)
E76B  23           inc hl
E76C  FE 1D        cp 1D            <next ADDRESS>
E76E  C8           ret z
E76F  FE 1E        cp 1E            <next LINE#>
E771  C2 EA E8     jp nz,E8EA       Error: Syntax error
E774  E5           push hl
E775  CD D6 DD      call DDD6       get BASIC line# at PC in <hl>, =carry
E778  DC B8 FF      call c,FFB8     test HL=DE? (try hl-de)
E77B  30 09          jr nc,E786     it's below PC, search from start
E77D  E1           pop hl
E77E  E5           push hl
E77F  CD F3 E8      call E8F3       command: ELSE ' REM
```

```
E782  23           inc hl
E783  CD A7 E7     call E7A7         search line# <de>, <hl>=address, =carry
E786  D4 9A E7     call nc,E79A      search line# <de> from start, <hl>=addr, nc=
E789  2B           dec hl
E78A  EB           ex de,hl
E78B  E1          pop hl
E78C  E5          push hl
E78D  3E 1D        ld a,1D           <next ADDRESS>
E78F  32 3A AE     ld (AE3A),a       BASIC Program line format
E792  2B           dec hl
E793  72           ld (hl),d
E794  2B           dec hl
E795  73           ld (hl),e
E796  2B           dec hl
E797  77           ld (hl),a
E798  E1          pop hl
E799  C9          ret

----- search line# <de> from start, <hl>=addr, nc=error
      @ C059! CBF0! DCDF! E708! E786! EAA1!
E79A  CD A3 E7    call E7A3          search line# <de> from start, <hl>=address,
E79D  D8          ret c
E79E  1E 08       ld e,08            Line does not exist
E7A0  C3 94 CA    jp CA94            perform ERROR <e> routine

----- search line# <de> from start, <hl>=address, =carry
      @ C10A! E110! E6E7! E741! E79A! E80A! E80F! E872!
E7A3  2A 81 AE    ld hl,(AE81)       start of BASIC program -1 pointer
E7A6  23          inc hl

----- search line# <de>, <hl>=address, =carry
      @ E783! E7BF'
E7A7  4E          ld c,(hl)
E7A8  23          inc hl
E7A9  46          ld b,(hl)          bc=len
E7AA  2B          dec hl
E7AB  78          ld a,b
E7AC  B1          or c
E7AD  C8          ret z              if bc=0, end of program
E7AE  E5          push hl
E7AF  23           inc hl
E7B0  23           inc hl
E7B1  7E           ld a,(hl)
E7B2  23           inc hl
E7B3  66           ld h,(hl)         <hl> = line#
E7B4  6F           ld l,a
E7B5  EB           ex de,hl
E7B6  CD B8 FF     call FFB8         test HL=DE? (try hl-de)
E7B9  EB           ex de,hl
E7BA  E1          pop hl
E7BB  3F          ccf
E7BC  D0          ret nc             return with (hl)=address
E7BD  C8          ret z
E7BE  09          add hl,bc          add len of this line = new address
E7BF  18 E6       jr E7A7            search line# <de>, <hl>=address, =carry

----- search line# <de>; <hl>=address
      @ E73C!
E7C1  2A 81 AE    ld hl,(AE81)       start of BASIC program -1 pointer
E7C4  23          inc hl
E7C5  E5          push hl
E7C6  4E           ld c,(hl)         <bc>=len of this line
E7C7  23           inc hl
E7C8  46           ld b,(hl)
E7C9  23           inc hl
```

```
E7CA  78            ld a,b
E7CB  B1            or c
E7CC  28 0F         jr z,E7DD        last line, len was 0
E7CE  7E            ld a,(hl)
E7CF  23            inc hl
E7D0  66            ld h,(hl)        <hl> = line#
E7D1  6F            ld l,a
E7D2  EB            ex de,hl
E7D3  CD B8 FF      call FFB8        test HL=DE? (try hl-de)
E7D6  EB            ex de,hl
E7D7  38 04         jr c,E7DD        line# found > or = ;return
E7D9  E1           pop hl
E7DA  09           add hl,bc
E7DB  18 E8        jr E7C5           next line

E7DD  E1           pop hl
E7DE  C9           ret

----- command: RENUM [<line#>][,[<old line#>][,<step>]]
      @ DE8D
E7DF  11 0A 00     ld de,000A        default start = 10.
E7E2  28 05        jr z,E7E9         no argument given
E7E4  FE 2C        cp 2C             ',
E7E6  C4 E1 CE     call nz,CEE1      get line# into <de>
E7E9  D5           push de
E7EA  11 00 00      ld de,0000       default old line# =0000
E7ED  CD 55 DD      call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
E7F0  30 05         jr nc,E7F7       old line# argument not present
E7F2  FE 2C         cp 2C            ',
E7F4  C4 E1 CE      call nz,CEE1     get line# into <de>
E7F7  D5            push de
E7F8  11 0A 00       ld de,000A      default step = 10.
E7FB  CD 55 DD       call DD55       CHRBACK comma?; if=:CHRGET <a>, scf
E7FE  DC E1 CE       call c,CEE1     get line# into <de>
E801  CD 4A DD       call DD4A       CHRGOT <a>; end of statement? else syntax er
E804  E1            pop hl
E805  EB            ex de,hl
E806  E3            ex (sp),hl
E807  EB            ex de,hl
E808  D5            push de
E809  E5             push hl
E80A  CD A3 E7         call E7A3     search line# <de> from start, <hl>=address,
E80D  D1             pop de
E80E  E5             push hl
E80F  CD A3 E7         call E7A3     search line# <de> from start, <hl>=address,
E812  EB               ex de,hl
E813  E1             pop hl
E814  CD B8 FF       call FFB8       test HL=DE? (try hl-de)
E817  DA 55 E7       jp c,E755       Error: Improper argument
E81A  EB             ex de,hl
E81B  D1            pop de
E81C  C1           pop bc
E81D  D5           push de
E81E  E5            push hl
E81F  C5             push bc
E820  4E               ld c,(hl)
E821  23               inc hl
E822  46               ld b,(hl)
E823  78               ld a,b
E824  B1               or c
E825  28 13            jr z,E83A     ...
E827  2B               dec hl
E828  09               add hl,bc
E829  7E               ld a,(hl)
E82A  23               inc hl
```

```
E82B  B6           or (hl)
E82C  28 0C        jr z,E83A       ...
E82E  2B           dec hl
E82F  C1          pop bc
E830  E5          push hl
E831  EB           ex de,hl
E832  09           add hl,bc
E833  EB           ex de,hl
E834  DA 55 E7     jp c,E755       Error: Improper argument
E837  E1          pop hl
E838  18 E5       jr E81F          ...

E83A  01 64 E8     ld bc,E864      search for line# within text
E83D  CD FF E8     call E8FF       go thru Basic program and do function <bc>
E840  C1          pop bc
E841  E1         pop hl
E842  D1        pop de
E843  C5        push bc
E844  E5         push hl
E845  4E          ld c,(hl)
E846  23          inc hl
E847  46          ld b,(hl)
E848  23          inc hl
E849  78          ld a,b
E84A  B1          or c
E84B  28 0C       jr z,E859        ...
E84D  73          ld (hl),e
E84E  23          inc hl
E84F  72          ld (hl),d
E850  23          inc hl
E851  E1         pop hl
E852  09         add hl,bc
E853  C1        pop bc
E854  EB        ex de,hl
E855  09        add hl,bc
E856  EB        ex de,hl
E857  18 EA     jr E843            ...

E859  E1         pop hl
E85A  E1        pop hl
E85B  01 88 E8  ld bc,E888         find line# element; if not defined: error
E85E  CD FF E8  call E8FF          go thru Basic program and do function <bc>
E861  C3 64 C0  jp C064            reset Basic

----- search for line# within text
      @ E83A: E86C' E886'
E864  CD 43 E9  call E943          skip over a statement element
E867  FE 02     cp 02              end of statement?
E869  D8        ret c
E86A  FE 1E     cp 1E              <next LINE#>
E86C  20 F6     jr nz,E864         search for line# within text
E86E  E5        push hl
E86F  56         ld d,(hl)
E870  2B         dec hl
E871  5E         ld e,(hl)
E872  CD A3 E7   call E7A3         search line# <de> from start, <hl>=address,
E875  30 0E      jr nc,E885        not found
E877  2B         dec hl
E878  EB         ex de,hl
E879  E1        pop hl
E87A  E5        push hl
E87B  72         ld (hl),d
E87C  2B         dec hl
E87D  73         ld (hl),e
E87E  2B         dec hl
```

```
E87F  3E 1D       ld a,1D          <next ADDRESS>
E881  77          ld (hl),a
E882  32 3A AE    ld (AE3A),a      BASIC Program line format
E885  E1          pop hl
E886  18 DC       jr E864          search for line# within text

----- find line# element; if not defined: error
      @ E85B: E890' E89D'
E888  CD 43 E9    call E943        skip over a statement element
E88B  FE 02       cp 02            is it end of statement or line?
E88D  D8          ret c
E88E  FE 1E       cp 1E            <next LINE#>
E890  20 F6       jr nz,E888       find line# element; if not defined: error
E892  E5          push hl
E893  56          ld d,(hl)
E894  2B          dec hl
E895  5E          ld e,(hl)
E896  CD D6 DD    call DDD6        get BASIC line# at PC in <hl>, =carry
E899  CD 18 CB    call CB18        print 'undefined line <line#> in <line#>
E89C  E1          pop hl
E89D  18 E9       jr E888          find line# element; if not defined: error

----- skip over statements till [ELSE] or <line end>
      @ C6D7!
E89F  06 01       ld b,01          initial count for [IF]
E8A1  2B          dec hl

----- skip statements, count IF's and ELSE's
      @ E8AD' E8B0' E8BA'
E8A2  CD 43 E9    call E943        skip over a statement element
E8A5  B7          or a
E8A6  C8          ret z
E8A7  FE 01       cp 01            <statement end>
E8A9  28 07       jr z,E8B2        end of statement
E8AB  FE A1       cp A1            [IF]
E8AD  20 F3       jr nz,E8A2       skip statements, count IF's and ELSE's
E8AF  04          inc b
E8B0  18 F0       jr E8A2          skip statements, count IF's and ELSE's

E8B2  CD 43 E9    call E943        skip over a statement element
E8B5  FE 97       cp 97            [ELSE]
E8B7  20 EC       jr nz,E8A5       was not [ELSE], try next
E8B9  05          dec b
E8BA  20 E6       jr nz,E8A2       skip statements, count IF's and ELSE's
E8BC  CD 3F DD    call DD3F        CHRGET <a>, skip blank, cp 01
E8BF  04          inc b
E8C0  C9          ret

----- check if parentheses pair
      @ D6D9!
E8C1  7E          ld a,(hl)
E8C2  FE 5B       cp 5B            '[
E8C4  28 03       jr z,E8C9
E8C6  FE 28       cp 28            '(
E8C8  C0          ret nz
E8C9  06 00       ld b,00          init counter =0
E8CB  04          inc b
E8CC  CD 43 E9    call E943        skip over a statement element
E8CF  FE 5B       cp 5B            '[
E8D1  28 F8       jr z,E8CB
E8D3  FE 28       cp 28            '(
E8D5  28 F4       jr z,E8CB
E8D7  FE 5D       cp 5D            ']
E8D9  28 0A       jr z,E8E5
E8DB  FE 29       cp 29            ')
```

```
E8DD  28 06       jr z,E8E5
E8DF  FE 02       cp 02             'STX (^B)
E8E1  38 07       jr c,E8EA         Error: Syntax error
E8E3  18 E7       jr E8CC           continue search

E8E5  05          dec b
E8E6  20 E4       jr nz,E8CC
E8E8  23          inc hl
E8E9  C9          ret

----- Error: Syntax error
E8EA  1E 02       ld e,02           Syntax error
E8EC  C3 94 CA    jp CA94           perform ERROR <e> routine

----- command: DATA <list of<data>>   (skip this line)
      @ C6F0! CC28  D12D  DD1B! DE19
E8EF  06 01       ld b,01           skip till end of line
E8F1  18 02       jr E8F5

----- command: ELSE ' REM
      @ DE2F  DE81  DE8B  E77F!
E8F3  06 00       ld b,00           skip one element
E8F5  2B          dec hl
E8F6  CD 43 E9    call E943         skip over a statement element
E8F9  B7          or a
E8FA  C8          ret z
E8FB  B8          cp b
E8FC  20 F8       jr nz,E8F6        take next element
E8FE  C9          ret

----- go thru Basic program and do function <bc>
      @ E692! E83D! E85E! E98F!
E8FF  CD D2 DD    call DDD2         get BASIC program counter in <hl>
E902  E5          push hl
E903  2A 81 AE     ld hl,(AE81)     start of BASIC program -1 pointer
E906  23           inc hl
E907  7E           ld a,(hl)
E908  23           inc hl           check len of line
E909  B6           or (hl)
E90A  28 13        jr z,E91F        end of program
E90C  23           inc hl
E90D  CD CE DD     call DDCE        set BASIC program counter to <hl>
E910  23           inc hl
E911  C5           push bc
E912  CD F9 FF      call FFF9       jp(bc)
E915  C1           pop bc
E916  2B           dec hl
E917  CD 35 E9     call E935        test for ELSE or THEN in this line; z=found
E91A  B7           or a
E91B  20 F4        jr nz,E911       statement ended, next statement
E91D  18 E7        jr E906          line ended, get next line

E91F  E1          pop hl
E920  C3 CE DD    jp DDCE           set BASIC program counter to <hl>

----- test line for ELSE or THEN; error if pgm ended
      @ C9CF! CA23!
E923  CD 35 E9    call E935         test for ELSE or THEN in this line; z=found
E926  B7          or a
E927  C0          ret nz            statement, not line ended; return
E928  23          inc hl
E929  7E          ld a,(hl)
E92A  23          inc hl
E92B  B6          or (hl)           end of program?
E92C  59          ld e,c            either NEXT or WEND missing
```

```
E92D  CA 94 CA    jp z,CA94          perform ERROR <e> routine
E930  23          inc hl
E931  54          ld d,h
E932  5D          ld e,l
E933  23          inc hl
E934  C9          ret

----- test for ELSE or THEN in this line; z=found
      @ E917! E923! E940'
E935  CD 43 E9    call E943          skip over a statement element
E938  FE 02       cp 02              end of statement?
E93A  D8          ret c
E93B  FE 97       cp 97              [ELSE]
E93D  C8          ret z
E93E  FE EB       cp EB              [THEN]
E940  20 F3       jr nz,E935         test for ELSE or THEN in this line; z=found
E942  C9          ret

----- skip over a statement element
      @ E69D! E864! E888! E8A2! E8B2! E8CC! E8F6! E935! E997!
E943  CD 3F DD    call DD3F          CHRGET <a>, skip blank, cp 01
E946  C8          ret z              end of line?
E947  FE 0E       cp 0E              <const 0>
E949  38 1D       jr c,E968          skip over variable
E94B  FE 20       cp 20              'SPACE
E94D  38 29       jr c,E978          skip over constant
E94F  FE 22       cp 22              '"
E951  28 09       jr z,E95C          skip over "text"
E953  FE 7C       cp 7C              [TEST]
E955  28 19       jr z,E970          skip over EXTERNAL COMMAND
E957  FE FF       cp FF              [TOKEN SWITCH]
E959  C0          ret nz
E95A  23          inc hl
E95B  C9          ret

----- skip over "text"
E95C  23          inc hl
E95D  7E          ld a,(hl)
E95E  FE 22       cp 22              '"
E960  C8          ret z
E961  B7          or a
E962  20 F8       jr nz,E95C         skip over "text"
E964  2B          dec hl
E965  3E 22       ld a,22            '"
E967  C9          ret

----- skip over variable
E968  FE 08       cp 08              'BS  (^H)
E96A  C8          ret z
E96B  FE 07       cp 07              'BEL (^G)
E96D  C8          ret z
E96E  23          inc hl
E96F  23          inc hl

----- skip over EXTERNAL COMMAND
E970  F5          push af
E971  23           inc hl
E972  7E           ld a,(hl)
E973  17           rla
E974  30 FB        jr nc,E971        next
E976  F1          pop af
E977  C9          ret
```

```
----- skip over constant
E978  FE 18        cp 18              'CAN (^X)
E97A  D8           ret c
E97B  FE 19        cp 19              <next byte VAL>
E97D  28 08        jr z,E987
E97F  FE 1F        cp 1F              <next 5 byte REAL>
E981  38 03        jr c,E986
E983  23           inc hl
E984  23           inc hl
E985  23           inc hl
E986  23           inc hl
E987  23           inc hl
E988  C9           ret

----- clear all VARIABLE indices
      @ C197! D9C0! EA74! EC47!
E989  C5           push bc
E98A  D5            push de
E98B  E5             push hl
E98C  01 96 E9        ld bc,E996      find any variable and clear index
E98F  CD FF E8        call E8FF       go thru Basic program and do function <bc>
E992  E1             pop hl
E993  D1            pop de
E994  C1           pop bc
E995  C9           ret

----- find any variable and clear index
      @ E98C: E9A0' E9A4' E9A8' E9BB'
E996  E5           push hl
E997  CD 43 E9      call E943         skip over a statement element
E99A  D1           pop de
E99B  FE 02        cp 02              end of statement?
E99D  D8           ret c
E99E  FE 0E        cp 0E              is it a variable?
E9A0  30 F4        jr nc,E996         find any variable and clear index
E9A2  FE 07        cp 07              'BEL (^G)
E9A4  28 F0        jr z,E996          find any variable and clear index
E9A6  FE 08        cp 08              'BS  (^H)
E9A8  28 EC        jr z,E996          find any variable and clear index
E9AA  EB           ex de,hl
E9AB  CD 3F DD     call DD3F          CHRGET <a>, skip blank, cp 01
E9AE  FE 0D        cp 0D              <real var>
E9B0  38 02        jr c,E9B4          ...
E9B2  36 0D        ld (hl),0D         <real var>
E9B4  23           inc hl
E9B5  36 00        ld (hl),00         clear VARIABLE table index
E9B7  23           inc hl
E9B8  36 00        ld (hl),00         clear VARIABLE table index
E9BA  EB           ex de,hl
E9BB  18 D9        jr E996            find any variable and clear index

----- command: RUN [<line#>]
      @ DE95
E9BD  CD 51 DD     call DD51          CHRGOT <a>; end of statement? =carry
E9C0  EB           ex de,hl
E9C1  2A 81 AE     ld hl,(AE81)       start of BASIC program -1 pointer
E9C4  EB           ex de,hl
E9C5  38 1C        jr c,E9E3          no line# given
E9C7  FE 1E        cp 1E              <next LINE#>
E9C9  28 15        jr z,E9E0          get line# and RUN
E9CB  FE 1D        cp 1D              <next ADDRESS>
E9CD  28 11        jr z,E9E0          get line# and RUN
E9CF  CD 0D EA     call EA0D          get filename, OPENIN, get startaddr, NEW
E9D2  21 30 EA     ld hl,EA30         load a file into store
E9D5  D2 13 BD     jp nc,BD13         MC BOOT PROGRAM, load and run FOREGROUND
```

```
E9D8  CD A8 EB    call EBA8         test filetype and load inputfile
E9DB  2A 81 AE    ld hl,(AE81)      start of BASIC program -1 pointer
E9DE  18 11       jr E9F1

----- get line# and RUN
E9E0  CD 67 E7    call E767         get address VAL into <de>
E9E3  D5          push de
E9E4  CD AD D2     call D2AD        CAS in/out abandon, release I/O buffers
E9E7  CD 8C C1     call C18C        reset all VARIABLE pointers
E9EA  CD 7A C1     call C17A        reset pointers; program, error, data
E9ED  CD 5E C1     call C15E        reset to RAD
E9F0  E1          pop hl
E9F1  23          inc hl
E9F2  F1          pop af
E9F3  C3 93 DD    jp DD93           RUN  LOOP, part 2

----- command: LOAD <filename>[,<startaddress>]
      @ DE51
E9F6  CD 0D EA    call EA0D         get filename, OPENIN, get startaddr, NEW
E9F9  30 06       jr nc,EA01        ...
E9FB  CD A8 EB    call EBA8         test filetype and load inputfile
E9FE  C3 64 C0    jp C064           reset Basic

EA01  E5          push hl
EA02  CD 01 F5     call F501        enough space in memory for <bc>?
EA05  CD 30 EA     call EA30        load a file into store
EA08  CA 6B CB     jp z,CB6B        perform a BREAK
EA0B  E1          pop hl
EA0C  C9          ret

----- get filename, OPENIN, get startaddr, NEW
      @ E9CF! E9F6!
EA0D  CD 8F EB    call EB8F         release I/O; get <filename>; OPENIN
EA10  E6 0E       and 0E            mask out
EA12  EE 02       xor 02            <integer VAR%>
EA14  28 0B       jr z,EA21         set loadpointer to <startaddr>, if present
EA16  CD 4A DD    call DD4A         CHRGOT <a>; end of statement? else syntax er
EA19  CD 8C C1    call C18C         reset all VARIABLE pointers
EA1C  CD 6B C1    call C16B         performs NEW, part 2
EA1F  37          scf
EA20  C9          ret

----- set loadpointer to <startaddr>, if present
EA21  CD 55 DD    call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
EA24  DC 91 CE    call c,CE91       get unsigned-integer VAL(expr) in <de>
EA27  ED 53 3F AE ld (AE3F),de      load pointer while LOAD
EA2B  CD 4A DD    call DD4A         CHRGOT <a>; end of statement? else syntax er
EA2E  B7          or a
EA2F  C9          ret

----- load a file into store
      @ E9D2: EA05!
EA30  2A 3F AE    ld hl,(AE3F)      load pointer while LOAD
EA33  CD 83 BC    call BC83         CAS IN DIRECT, read input file into store (h
EA36  E5          push hl
EA37  DC 7A BC     call c,BC7A      CAS IN CLOSE
EA3A  E1          pop hl
EA3B  C9          ret

----- command: CHAIN <filename>[,<run line#>] [,DELETE<line#>[-<line#>]]
      @ DE0B
EA3C  EE AB       xor AB            [MERGE]
EA3E  20 04       jr nz,EA44        no
EA40  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
EA43  37          scf
```

```
EA44  9F          sbc a,a            0 = load, FF = merge
EA45  32 41 AE    ld (AE41),a        LOAD/MERGE flag
EA48  CD 8F EB    call EB8F          release I/O; get <filename>; OPENIN
EA4B  11 00 00    ld de,0000         default line# 0000
EA4E  CD 55 DD    call DD55          CHRBACK comma?; if=:CHRGET <a>, scf
EA51  30 06       jr nc,EA59         <line#> argument not present, take default
EA53  7E          ld a,(hl)
EA54  FE 2C       cp 2C              ','
EA56  C4 91 CE    call nz,CE91       get unsigned-integer VAL(expr) in <de>
EA59  D5          push de
EA5A  CD 55 DD     call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
EA5D  3E 00        ld a,00
EA5F  30 09        jr nc,EA6A        ...
EA61  CD 37 DD     call DD37         CHRNEXT <a>, nz=Error; CHRGET
EA64  92           [DELETE]
EA65  CD 37 E7     call E737         get bounds of the lines to delete
EA68  3E FF        ld a,FF
EA6A  F5           push af
EA6B  CD 4A DD      call DD4A        CHRGOT <a>; end of statement? else syntax er
EA6E  CD 1B FB      call FB1B        ...
EA71  CD 3E FC      call FC3E        GARBAGE COLLECT
EA74  CD 89 E9      call E989        clear all VARIABLE indices
EA77  CD D2 D5      call D5D2        clear AE04..5 to 0
EA7A  CD 49 F5      call F549        ...
EA7D  F1           pop af
EA7E  C5           push bc
EA7F  D5            push de
EA80  B7             or a
EA81  C4 5A E7       call nz,E75A    delete line area; shift rest of pgm down
EA84  3A 41 AE       ld a,(AE41)     LOAD/MERGE flag
EA87  B7             or a
EA88  20 08          jr nz,EA92      merge
EA8A  CD 6B C1       call C16B       performs NEW, part 2
EA8D  CD A8 EB       call EBA8       test filetype and load inputfile
EA90  18 03          jr EA95

EA92  CD 9D EB       call EB9D       ...
EA95  D1            pop de
EA96  C1           pop bc
EA97  CD 71 F5     call F571         ...
EA9A  D1          pop de
EA9B  2A 81 AE    ld hl,(AE81)       start of BASIC program -1 pointer
EA9E  7A          ld a,d
EA9F  B3          or e
EAA0  C8          ret z
EAA1  CD 9A E7    call E79A          search line# <de> from start, <hl>=addr, nc=
EAA4  2B          dec hl
EAA5  C9          ret

----- command: MERGE [<filename>]
      @ DE57
EAA6  CD 8F EB    call EB8F          release I/O; get <filename>; OPENIN
EAA9  CD 4A DD    call DD4A          CHRGOT <a>; end of statement? else syntax er
EAAC  CD 8C C1    call C18C          reset all VARIABLE pointers
EAAF  CD 9D EB    call EB9D          ...
EAB2  C3 64 C0    jp C064            reset Basic

      @ EBA1
EAB5  CD 7A C1    call C17A          reset pointers; program, error, data
EAB8  CD 87 E6    call E687          change Basic program to line# format
EABB  2A 83 AE    ld hl,(AE83)       end of BASIC program pointer
EABE  EB          ex de,hl
EABF  2A 81 AE    ld hl,(AE81)       start of BASIC program -1 pointer
EAC2  23          inc hl
EAC3  22 83 AE    ld (AE83),hl       end of BASIC program pointer
```

```
EAC6  EB        ex de,hl
EAC7  CD DA FF  call FFDA      BC=HL-DE
EACA  EB        ex de,hl
EACB  2A 8D B0  ld hl,(B08D)   low end of used string space pointer
EACE  EB        ex de,hl
EACF  2B        dec hl
EAD0  CD F5 FF  call FFF5      lddr
EAD3  13        inc de
EAD4  EB        ex de,hl

      @ EB0C' EB18'
EAD5  E5        push hl
EAD6  2A 83 AE   ld hl,(AE83)  end of BASIC program pointer
EAD9  11 20 00   ld de,0020
EADC  19         add hl,de
EADD  EB         ex de,hl
EADE  E1        pop hl
EADF  CD B8 FF  call FFB8      test HL=DE? (try hl-de)
EAE2  38 50     jr c,EB34      Error: Memory full
EAE4  CD 84 EB  call EB84      ...
EAE7  B3        or e
EAE8  28 30     jr z,EB1A      ...
EAEA  D5        push de
EAEB  CD 84 EB   call EB84     ...

      @ EB04'
EAEE  E5        push hl
EAEF  7E         ld a,(hl)
EAF0  23         inc hl
EAF1  B6         or (hl)
EAF2  28 12      jr z,EB06     ...
EAF4  23         inc hl
EAF5  7E         ld a,(hl)
EAF6  23         inc hl
EAF7  66         ld h,(hl)
EAF8  6F         ld l,a
EAF9  CD B8 FF   call FFB8     test HL=DE? (try hl-de)
EAFC  E1        pop hl
EAFD  28 0F     jr z,EB0E      ...
EAFF  30 06     jr nc,EB07     ...
EB01  CD 48 EB  call EB48      ...
EB04  18 E8     jr EAEE        ...

EB06  E1        pop hl
EB07  E3        ex (sp),hl
EB08  CD 5E EB  call EB5E      ...
EB0B  E1        pop hl
EB0C  18 C7     jr EAD5        ...

EB0E  E3        ex (sp),hl
EB0F  CD 5E EB  call EB5E      ...
EB12  E1        pop hl
EB13  5E        ld e,(hl)
EB14  23        inc hl
EB15  56        ld d,(hl)
EB16  2B        dec hl
EB17  19        add hl,de
EB18  18 BB     jr EAD5        ...

      @ EAE8' EB23'
EB1A  7E        ld a,(hl)
EB1B  23        inc hl
EB1C  B6        or (hl)
EB1D  2B        dec hl
EB1E  28 05     jr z,EB25      ...
```

```
EB20  CD 48 EB    call EB48          ...
EB23  18 F5       jr EB1A            ...

      @ EB1E'
EB25  2A 83 AE    ld hl,(AE83)       end of BASIC program pointer
EB28  36 00       ld (hl),00
EB2A  23          inc hl
EB2B  36 00       ld (hl),00
EB2D  23          inc hl
EB2E  22 83 AE    ld (AE83),hl       end of BASIC program pointer
EB31  C3 B1 D5    jp D5B1            reset all VARIABLE pointers to basic start

----- Error: Memory full
      @ EAE2' EBD1
EB34  1E 07       ld e,07            Memory full
EB36  18 02       jr EB3A

----- Error: EOF met
      @ EB7A' EB8B' EBE6
EB38  1E 18       ld e,18            EOF met
EB3A  D5          push de
EB3B  CD AD D2     call D2AD         CAS in/out abandon, release I/O buffers
EB3E  CD 8C C1     call C18C         reset all VARIABLE pointers
EB41  CD 6B C1     call C16B         performs NEW, part 2
EB44  D1          pop de
EB45  C3 94 CA    jp CA94            perform ERROR <e> routine

      @ EB01! EB20!
EB48  C5          push bc
EB49  D5           push de
EB4A  E5            push hl
EB4B  4E             ld c,(hl)
EB4C  23             inc hl
EB4D  46             ld b,(hl)
EB4E  2A 83 AE       ld hl,(AE83)    end of BASIC program pointer
EB51  EB             ex de,hl
EB52  E1            pop hl
EB53  CD F2 FF      call FFF2        ldir
EB56  EB            ex de,hl
EB57  22 83 AE      ld (AE83),hl     end of BASIC program pointer
EB5A  EB            ex de,hl
EB5B  D1           pop de
EB5C  C1          pop bc
EB5D  C9          ret

      @ EB08! EB0F!
EB5E  D5          push de
EB5F  EB           ex de,hl
EB60  2A 83 AE     ld hl,(AE83)      end of BASIC program pointer
EB63  73           ld (hl),e
EB64  23           inc hl
EB65  72           ld (hl),d
EB66  23           inc hl
EB67  EB           ex de,hl
EB68  E3           ex (sp),hl
EB69  EB           ex de,hl
EB6A  73           ld (hl),e
EB6B  23           inc hl
EB6C  72           ld (hl),d
EB6D  23           inc hl
EB6E  D1          pop de
EB6F  1B          dec de
EB70  1B          dec de
EB71  1B          dec de
```

```
      @ EB7E'
EB72  1B           dec de
EB73  7A           ld a,d
EB74  B3           or e
EB75  28 09        jr z,EB80         ...
EB77  CD 80 BC     call BC80         CAS IN CHAR from input file
EB7A  30 BC        jr nc,EB38        Error: EOF met
EB7C  77           ld (hl),a
EB7D  23           inc hl
EB7E  18 F2        jr EB72           ...

      @ EB75'
EB80  22 83 AE     ld (AE83),hl      end of BASIC program pointer
EB83  C9           ret

      @ EAE4! EAEB!
EB84  CD 80 BC     call BC80         CAS IN CHAR from input file
EB87  5F           ld e,a
EB88  DC 80 BC     call c,BC80       CAS IN CHAR from input file
EB8B  30 AB        jr nc,EB38        Error: EOF met
EB8D  57           ld d,a
EB8E  C9           ret

----- release I/O; get <filename>; OPENIN
      @ EA0D! EA48! EAA6!
EB8F  CD AD D2     call D2AD         CAS in/out abandon, release I/O buffers
EB92  CD 6A D2     call D26A         get <filename>, allocate buff, OPENIN
EB95  32 42 AE     ld (AE42),a       LOAD/CHAIN flag
EB98  ED 43 43 AE ld (AE43),bc       used by LOAD, CHAIN
EB9C  C9           ret

      @ EA92! EAAF!
EB9D  3A 42 AE     ld a,(AE42)       LOAD/CHAIN flag
EBA0  B7           or a
EBA1  CA B5 EA     jp z,EAB5         ...
EBA4  FE 16        cp 16             filetpe
EBA6  20 0B        jr nz,EBB3        Error: File type error

----- test filetype and load inputfile
      @ E9D8! E9FB! EA8D!
EBA8  3A 42 AE     ld a,(AE42)       LOAD/CHAIN flag
EBAB  FE 16        cp 16             filetype
EBAD  28 40        jr z,EBEF         load as Ascii file
EBAF  E6 FE        and FE            =254.
EBB1  28 05        jr z,EBB8         load as a Basic program

----- Error: File type error
EBB3  1E 19        ld e,19           File type error
EBB5  C3 94 CA     jp CA94           perform ERROR <e> routine

----- load as a Basic program
EBB8  CD 7A C1     call C17A         reset pointers; program, error, data
EBBB  2A 81 AE     ld hl,(AE81)      start of BASIC program -1 pointer
EBBE  23           inc hl
EBBF  EB           ex de,hl
EBC0  2A 8D B0     ld hl,(B08D)      low end of used string space pointer
EBC3  01 80 FF     ld bc,FF80        = -80
EBC6  09           add hl,bc
EBC7  ED 4B 43 AE ld bc,(AE43)       used by LOAD, CHAIN
EBCB  CD CF FF     call FFCF         HL=HL-DE
EBCE  D4 BE FF     call nc,FFBE      test HL=BC? (try hl-bc)
EBD1  DA 34 EB     jp c,EB34         Error: Memory full
EBD4  60           ld h,b
EBD5  69           ld l,c
EBD6  19           add hl,de
```

```
EBD7  22 83 AE    ld (AE83),hl       end of BASIC program pointer
EBDA  3A 42 AE    ld a,(AE42)        LOAD/CHAIN flag
EBDD  1F          rra
EBDE  9F          sbc a,a
EBDF  32 45 AE    ld (AE45),a        flag file read protected
EBE2  EB          ex de,hl
EBE3  CD 83 BC    call BC83          CAS IN DIRECT, read input file into store (h
EBE6  CA 38 EB    jp z,EB38          Error: EOF met
EBE9  CD B1 D5    call D5B1          reset all VARIABLE pointers to basic start
EBEC  C3 98 D2    jp D298            command: CLOSEIN

----- load as Ascii file
      @ EBAD'
EBEF  CD 7A C1    call C17A          reset pointers; program, error, data
EBF2  CD CB DD    call DDCB          reset BASIC program counter
EBF5  CD 4C CA    call CA4C          read a line from tape to edit buffer
EBF8  D2 98 D2    jp nc,D298         command: CLOSEIN
EBFB  CD BC E6    call E6BC          check for a line# in command line
EBFE  38 F5       jr c,EBF5          next line
EC00  1E 15       ld e,15            Direct command found
EC02  28 02       jr z,EC06          yes!
EC04  1E 06       ld e,06            Overflow
EC06  C3 94 CA    jp CA94            perform ERROR <e> routine

----- command: SAVE <filename>[,<filetype>][,<startaddr>,<len>]
      @ DE97
EC09  CD AD D2    call D2AD          CAS in/out abandon, release I/O buffers
EC0C  CD 56 D2    call D256          command: OPENOUT <filename>
EC0F  06 00       ld b,00            default is savefile
EC11  CD 55 DD    call DD55          CHRBACK comma?; if=:CHRGET <a>, scf
EC14  30 29       jr nc,EC3F         save as savefile
EC16  CD 37 DD    call DD37          CHRNEXT <a>, nz=Error; CHRGET
EC19  0D          'CR  (^M)
EC1A  23          inc hl
EC1B  23          inc hl
EC1C  7E          ld a,(hl)
EC1D  23          inc hl
EC1E  E6 DF       and DF             mask out
EC20  F2 38 EC    jp p,EC38          Error: Syntax error
EC23  E5          push hl
EC24  21 2C EC     ld hl,EC2C        table of filetypes A B P
EC27  CD 93 FF     call FF93         search <a> in table(hl); hl=address
EC2A  E3           ex (sp),hl
EC2B  C9           ret               jump to found label

----- table of filetypes A B P
      @ EC24:
EC2C  03 38 EC    EC38 =3.           Error: Syntax error
EC2F  C1 87 EC    EC87 =193.         A, save as ASCII file
EC32  C2 5C EC    EC5C =194.         B, save as binary file
EC35  D0 3D EC    EC3D =208.         P, save as protected file

----- Error: Syntax error
EC38  1E 02       ld e,02            Syntax error
EC3A  C3 94 CA    jp CA94            perform ERROR <e> routine

----- P, save as protected file
      @ EC35:
EC3D  06 01       ld b,01            filetype protected
EC3F  CD 4A DD    call DD4A          CHRGOT <a>; end of statement? else syntax er
EC42  E5          push hl
EC43  C5           push bc
EC44  CD 87 E6      call E687        change Basic program to line# format
EC47  CD 89 E9      call E989        clear all VARIABLE indices
EC4A  2A 81 AE      ld hl,(AE81)     start of BASIC program -1 pointer
```

```
EC4D  23              inc hl
EC4E  EB              ex de,hl
EC4F  2A 83 AE        ld hl,(AE83)     end of BASIC program pointer
EC52  CD CF FF        call FFCF        HL=HL-DE
EC55  EB              ex de,hl
EC56  F1             pop af
EC57  01 00 00       ld bc,0000
EC5A  18 23          jr EC7F           write, close, return

----- B, save as binary file
      @ EC32:
EC5C  06 02         ld b,02            filetype binary
EC5E  CD 37 DD      call DD37          CHRNEXT <a>, nz=Error; CHRGET
EC61  2C            ',
EC62  CD 91 CE      call CE91          get unsigned-integer VAL(expr) in <de>
EC65  D5            push de
EC66  CD 37 DD       call DD37         CHRNEXT <a>, nz=Error; CHRGET
EC69  2C             ',
EC6A  CD 91 CE       call CE91         get unsigned-integer VAL(expr) in <de>
EC6D  D5             push de
EC6E  CD 55 DD        call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
EC71  11 00 00        ld de,0000
EC74  DC 91 CE        call c,CE91      get unsigned-integer VAL(expr) in <de>
EC77  D5              push de
EC78  CD 4A DD         call DD4A       CHRGOT <a>; end of statement? else syntax er
EC7B  78               ld a,b
EC7C  C1              pop bc
EC7D  D1             pop de
EC7E  E3             ex (sp),hl
EC7F  CD 98 BC       call BC98         CAS OUT DIRECT, (hl)=data, <de>=len, <a>=typ
EC82  D2 6B CB       jp nc,CB6B        perform a BREAK
EC85  18 17          jr EC9E           closeout, return

----- A, save as ASCII file
      @ EC2F:
EC87  CD 4A DD      call DD4A          CHRGOT <a>; end of statement? else syntax er
EC8A  E5            push hl
EC8B  3E 09          ld a,09           channel 9=tape
EC8D  CD A2 C1       call C1A2         set output channel to <a>, a= old channel
EC90  F5             push af
EC91  01 01 00        ld bc,0001       default start line#
EC94  11 FF FF        ld de,FFFF       default end line#
EC97  CD 0D E1        call E10D        perform LIST
EC9A  F1             pop af
EC9B  CD A2 C1       call C1A2         set output channel to <a>, a= old channel

----- closeout, return
EC9E  CD A1 D2       call D2A1         command: CLOSEOUT
ECA1  E1            pop hl
ECA2  C9            ret

----- get either HEX or integer VAL
      @ D571! DC16! FA87!
ECA3  CD 44 ED      call ED44          look for - (d=FF) or + (d=00)
ECA6  20 05         jr nz,ECAD         sign not present
ECA8  CD 61 DD      call DD61          CHRSKIP <a>; skip over blank, tab, linefeed
ECAB  18 2F         jr ECDC            get integer VAL

ECAD  FE 26         cp 26              '&
ECAF  28 1C         jr z,ECCD          get HEX VAL
ECB1  CD 7F FF      call FF7F          test <a> for '.' or digit, =CARRY
ECB4  38 26         jr c,ECDC          get integer VAL
ECB6  CD 10 FF      call FF10          set VARTYPE integer; <a>=2
ECB9  CD F3 FE      call FEF3          set FAC to all zeroes
ECBC  37            scf
```

```
ECBD  C9          ret

----- get either HEX or integer VAL
      @ E01C! E05C!
ECBE  E5          push hl
ECBF  CD C6 EC     call ECC6        do it here
ECC2  D1          pop de
ECC3  D8          ret c
ECC4  EB          ex de,hl
ECC5  C9          ret

----- do it here
ECC6  16 00       ld d,00          'NUL
ECC8  7E          ld a,(hl)
ECC9  FE 26       cp 26            '&
ECCB  20 0F       jr nz,ECDC       get integer VAL

----- get HEX VAL
ECCD  CD 1C EE    call EE1C        convert Ascii HEX or BIN to <hl>
ECD0  EB          ex de,hl
ECD1  F5          push af
ECD2  CD 0D FF     call FF0D        set FAC to <hl> and mark integer
ECD5  F1          pop af
ECD6  EB          ex de,hl
ECD7  D8          ret c
ECD8  C8          ret z
ECD9  C3 F3 CA    jp CAF3          Error: Overflow

----- get integer VAL
      @ ECAB' ECB4' ECCB'
ECDC  E5          push hl
ECDD  7E           ld a,(hl)
ECDE  23           inc hl
ECDF  FE 2E        cp 2E           '.
ECE1  CC 61 DD     call z,DD61     CHRSKIP <a>; skip over blank, tab, linefeed
ECE4  CD 83 FF     call FF83       test <a> for digit, =carry
ECE7  E1          pop hl
ECE8  38 06       jr c,ECF0        yes, a digit
ECEA  7E          ld a,(hl)
ECEB  EE 2E       xor 2E           '.
ECED  C0          ret nz
ECEE  23          inc hl
ECEF  C9          ret

      @ ECE8'
ECF0  CD 10 FF    call FF10        set VARTYPE integer; <a>=2
ECF3  D5          push de
ECF4  01 00 00     ld bc,0000
ECF7  11 46 AE     ld de,AE46      number edit buffer (1)
ECFA  CD 53 ED     call ED53       ...
ECFD  FE 2E        cp 2E           '.
ECFF  20 0B        jr nz,ED0C      ...
ED01  CD C9 ED     call EDC9       ...
ED04  CD 19 FF     call FF19       set VARTYPE real; <a>=5
ED07  0C           inc c
ED08  CD 53 ED     call ED53       ...
ED0B  0D           dec c
ED0C  F5           push af
ED0D  3E FF         ld a,FF        'IGNORE
ED0F  12            ld (de),a
ED10  F1           pop af
ED11  CD 77 ED     call ED77       ...
ED14  D1          pop de
ED15  5F          ld e,a
ED16  E5          push hl
```

```
ED17  D5          push de
ED18  21 46 AE     ld hl,AE46      number edit buffer (1)
ED1B  CD CE ED     call EDCE       ...
ED1E  D1          pop de
ED1F  CD 27 FF    call FF27       get VARTYPF <a>; cp string
ED22  30 08       jr nc,ED2C      ...
ED24  E5          push hl
ED25  42           ld b,d
ED26  CD 06 FE     call FE06       ...
ED29  E1          pop hl
ED2A  38 11       jr c,ED3D       ...
ED2C  7A          ld a,d
ED2D  4E          ld c,(hl)
ED2E  23          inc hl
ED2F  CD 94 BD    call BD94       REAL ARITH ??
ED32  7B          ld a,e
ED33  CD 55 BD    call BD55       REAL ARITH ??
ED36  EB          ex de,hl
ED37  CD 16 FF    call FF16       ld hl,FAC; set VARTYPE real; <a>=5
ED3A  DC 3D BD    call c,BD3D     copy 5 bytes,(de)>(hl); ld a,(hl-1)
ED3D  3E 0A       ld a,0A         'LF  (^J)
ED3F  E1         pop hl
ED40  D8         ret c
ED41  C3 F3 CA   jp CAF3          Error: Overflow

----- look for - (d=FF) or + (d=00)
      @ ECA3! ED7F!
ED44  CD 61 DD   call DD61        CHRSKIP <a>; skip over blank, tab, linefeed
ED47  23         inc hl
ED48  16 FF      ld d,FF          FF flag for -
ED4A  FE 2D      cp 2D            '-
ED4C  C8         ret z
ED4D  14         inc d            00 flag for +
ED4E  FE 2B      cp 2B            '+
ED50  C8         ret z
ED51  2B         dec hl
ED52  C9         ret

      @ ECFA! ED08! ED72' ED75'
ED53  E5         push hl
ED54  CD 61 DD    call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
ED57  23          inc hl
ED58  CD 83 FF    call FF83       test <a> for digit, =carry
ED5B  38 04       jr c,ED61       it's a digit
ED5D  E1         pop hl
ED5E  C3 8A FF   jp FF8A          change <a> to upper case

ED61  E3          ex (sp),hl
ED62  E1         pop hl
ED63  D6 30      sub 30           remove ascii part
ED65  12         ld (de),a
ED66  B0         or b
ED67  28 07      jr z,ED70        ...
ED69  78         ld a,b
ED6A  04         inc b
ED6B  FE 0C      cp 0C            'FF  (^L)
ED6D  30 01      jr nc,ED70       ...
ED6F  13         inc de
ED70  79         ld a,c
ED71  B7         or a
ED72  28 DF      jr z,ED53        ...
ED74  0C         inc c
ED75  18 DC      jr ED53          ...
```

```
     @ ED11!
ED77  FE 45       cp 45           'E
ED79  20 10       jr nz,ED8B      ...
ED7B  E5          push hl
ED7C  CD C9 ED     call EDC9      ...
ED7F  CD 44 ED     call ED44      look for - (d=FF) or + (d=00)
ED82  CC 61 DD     call z,DD61    CHRSKIP <a>; skip over blank, tab, linefeed
ED85  CD 83 FF     call FF83      test <a> for digit, =carry
ED88  38 04        jr c,ED8E      ...
ED8A  E1          pop hl
ED8B  AF          xor a
ED8C  18 1E       jr EDAC         ...

ED8E  E3           ex (sp),hl
ED8F  E1          pop hl
ED90  CD 19 FF    call FF19       set VARTYPE real; <a>=5
ED93  D5          push de
ED94  C5           push bc
ED95  CD 35 EE      call EE35     convert Ascii decimal# to <hl>
ED98  30 09         jr nc,EDA3    ...
ED9A  7B            ld a,e
ED9B  D6 64         sub 64        'd
ED9D  7A            ld a,d
ED9E  DE 00         sbc a,00      'NUL
EDA0  7B            ld a,e
EDA1  38 02         jr c,EDA5     ...
EDA3  3E 7F         ld a,7F       'DEL
EDA5  C1           pop bc
EDA6  D1          pop de
EDA7  14          inc d
EDA8  20 02       jr nz,EDAC      ...
EDAA  2F          cpl
EDAB  3C          inc a
EDAC  C6 80       add a,80        'F0  0
EDAE  5F          ld e,a
EDAF  78          ld a,b
EDB0  D6 0C       sub 0C          'FF  (^L)
EDB2  30 01       jr nc,EDB5      ...
EDB4  AF          xor a
EDB5  91          sub c
EDB6  30 09       jr nc,EDC1      ...
EDB8  83          add a,e
EDB9  38 01       jr c,EDBC       ...
EDBB  AF          xor a
EDBC  FE 01       cp 01           'SOH (^A)
EDBE  CE 80       adc a,80        'F0  0
EDC0  C9          ret

EDC1  83          add a,e
EDC2  30 02       jr nc,EDC6      ...
EDC4  3E FF       ld a,FF         'IGNORE
EDC6  D6 80       sub 80          'F0  0
EDC8  C9          ret

     @ ED01! ED7C!
EDC9  CD 61 DD    call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
EDCC  23          inc hl
EDCD  C9          ret

     @ ED1B!
EDCE  EB          ex de,hl
EDCF  21 58 AE    ld hl,AE58      number edit buffer (3)
EDD2  01 01 05    ld bc,0501      ...
EDD5  2B          dec hl
EDD6  36 00       ld (hl),00      'NUL
```

```
EDD8  10 FB      djnz EDD5       ...
EDDA  1A         ld a,(de)
EDDB  FE FF      cp FF           'IGNORE
EDDD  C8         ret z
EDDE  77         ld (hl),a
EDDF  21 53 AE   ld hl,AE53      number edit buffer (2)
EDE2  13         inc de
EDE3  1A         ld a,(de)
EDE4  FE FF      cp FF           'IGNORE
EDE6  C8         ret z
EDE7  D5         push de
EDE8  41          ld b,c
EDE9  16 00       ld d,00        'NUL
EDEB  E5          push hl
EDEC  5E           ld e,(hl)
EDED  62           ld h,d
EDEE  6B           ld l,e
EDEF  29           add hl,hl
EDF0  29           add hl,hl
EDF1  19           add hl,de
EDF2  29           add hl,hl
EDF3  5F           ld e,a
EDF4  19           add hl,de
EDF5  5D           ld e,l
EDF6  7C           ld a,h
EDF7  E1          pop hl
EDF8  73          ld (hl),e
EDF9  23          inc hl
EDFA  10 EF       djnz EDEB      ...
EDFC  D1         pop de
EDFD  B7         or a
EDFE  28 DF      jr z,EDDF       ...
EE00  77         ld (hl),a
EE01  0C         inc c
EE02  18 DB      jr EDDF         ...

      @ E011! E6C2!
EE04  C5         push bc
EE05  E5          push hl
EE06  CD 35 EE     call EE35     convert Ascii decimal# to <hl>
EE09  EB           ex de,hl
EE0A  CD 0D FF     call FF0D     set FAC to <hl> and mark integer
EE0D  EB           ex de,hl
EE0E  C1          pop bc
EE0F  30 06       jr nc,EE17     ...
EE11  7A          ld a,d
EE12  B3          or e
EE13  C6 FF       add a,FF       'IGNORE
EE15  38 03       jr c,EE1A      ...
EE17  50          ld d,b
EE18  59          ld e,c
EE19  EB          ex de,hl
EE1A  C1         pop bc
EE1B  C9         ret

----- convert Ascii HEX or BIN to <hl>
      @ ECCD!
EE1C  23         inc hl          skip over '&
EE1D  CD 61 DD   call DD61       CHRSKIP <a>; skip over blank, tab, linefeed
EE20  CD 8A FF   call FF8A       change <a> to upper case
EE23  06 02      ld b,02         binary number?
EE25  FE 58      cp 58           'X
EE27  28 06      jr z,EE2F       yes, binary
EE29  06 10      ld b,10         hex number?
EE2B  FE 48      cp 48           'H
```

```
EE2D  20 04      jr nz,EE33       none of these, must be hex
EE2F  23         inc hl           skip 'X or 'H
EE30  CD 61 DD   call DD61        CHRSKIP <a>; skip over blank, tab, linefeed
EE33  18 02      jr EE37

----- convert Ascii decimal# to <hl>
      @ ED95! EE06!
EE35  06 0A      ld b,0A          decimal number!
EE37  EB         ex de,hl
EE38  CD 61 EE   call EE61        a=VAL of digit or HEX digit
EE3B  26 00      ld h,00          'NUL
EE3D  6F         ld l,a
EE3E  30 1E      jr nc,EE5E       ...
EE40  0E 00      ld c,00          'NUL
EE42  CD 61 EE   call EE61        a=VAL of digit or HEX digit
EE45  30 14      jr nc,EE5B       no more digits; return
EE47  D5         push de
EE48  16 00       ld d,00         'NUL
EE4A  5F          ld e,a
EE4B  D5          push de
EE4C  58           ld e,b
EE4D  CD BE BD     call BDBE      INT ARITH, ??
EE50  D1          pop de
EE51  38 03       jr c,EE56       ...
EE53  19          add hl,de
EE54  30 02       jr nc,EE58      ...
EE56  0E FF       ld c,FF         'IGNORE
EE58  D1         pop de
EE59  18 E7      jr EE42          next digit

EE5B  79         ld a,c
EE5C  FE 01      cp 01
EE5E  EB         ex de,hl
EE5F  78         ld a,b
EE60  C9         ret

----- a=VAL of digit or HEX digit
      @ EE38! EE42!
EE61  1A         ld a,(de)
EE62  13         inc de
EE63  CD 83 FF   call FF83        test <a> for digit, =carry
EE66  38 0A      jr c,EE72        yes, digit
EE68  CD 8A FF   call FF8A        change <a> to upper case
EE6B  FE 41      cp 41            'A
EE6D  3F         ccf
EE6E  30 05      jr nc,EE75       no, not a HEX digit
EE70  D6 07      sub 07           remove HEX part
EE72  D6 30      sub 30           remove ascii part
EE74  B8         cp b
EE75  D8         ret c
EE76  1B         dec de
EE77  AF         xor a            =0
EE78  C9         ret

----- print line#
      @ C106! CB4C  DDF8!
EE79  CD 0D FF   call FF0D        set FAC to <hl> and mark integer
EE7C  CD 82 EE   call EE82        convert FAC to ASCII, (hl)=address of text
EE7F  C3 41 C3   jp C341          output text (hl) to channel

----- convert FAC to ASCII, (hl)=address of text
      @ E173! E29E! EE7C!
EE82  D5         push de
EE83  C5          push bc
EE84  CD C3 FC     call FCC3      ...
```

```
EE87  AF             xor a
EE88  CD A7 EE       call EEA7       ...
EE8B  23             inc hl
EE8C  C1             pop bc
EE8D  D1             pop de
EE8E  C9             ret

----- edit FAC onto the NUMBER EDIT BUFFER
      @ E2CF! F48E!
EE8F  D5             push de
EE90  C5             push bc
EE91  AF             xor a
EE92  CD 9F EE       call EE9F       ...
EE95  C1             pop bc
EE96  D1             pop de
EE97  7E             ld a,(hl)
EE98  FE 20          cp 20           'SPACE
EE9A  C0             ret nz
EE9B  23             inc hl
EE9C  C9             ret

      @ F23D! F91F!
EE9D  3E 40          ld a,40         '@

      @ EE92! F3B0! F916!
EE9F  22 6E AE       ld (AE6E),hl    temp store for char
EEA2  F5             push af
EEA3  CD B3 FC       call FCB3       ...
EEA6  F1             pop af

      @ EE88!
EEA7  C5             push bc
EEA8  57             ld d,a
EEA9  D5             push de
EEAA  EB             ex de,hl
EEAB  21 68 AE       ld hl,AE68      number edit buffer (4)
EEAE  36 00          ld (hl),00
EEB0  22 70 AE       ld (AE70),hl    number edit buffer index
EEB3  CD B7 F0       call F0B7       ...
EEB6  D1             pop de
EEB7  CD D4 EE       call EED4       ...
EEBA  CD 3D F0       call F03D       ...
EEBD  58             ld e,b
EEBE  C1             pop bc
EEBF  7B             ld a,e
EEC0  B7             or a
EEC1  CC 50 F0       call z,F050     ...
EEC4  CD 5F F0       call F05F       ...
EEC7  CD 69 F0       call F069       ...
EECA  CD 7C F0       call F07C       ...
EECD  7A             ld a,d
EECE  1F             rra
EECF  D0             ret nc
EED0  2B             dec hl
EED1  36 25          ld (hl),25      '%
EED3  C9             ret

      @ EEB7!
EED4  7A             ld a,d
EED5  87             add a,a
EED6  30 29          jr nc,EF01      ...
EED8  FA 27 EF       jp m,EF27       ...
EEDB  7B             ld a,e
EEDC  81             add a,c
EEDD  D6 0A          sub 0A          'LF  (^J)
```

```
EEDF  FA 88 EF    jp m,EF88        ...
EEE2  16 01       ld d,01          'SOH (^A)

      @ EF06' EF17
EEE4  41          ld b,c
EEE5  79          ld a,c
EEE6  B7          or a
EEE7  28 15       jr z,EEFE        ...
EEE9  83          add a,e
EEEA  3D          dec a
EEEB  5F          ld e,a
EEEC  CD 0E F0    call F00E        ...
EEEF  06 01       ld b,01          'SOH (^A)
EEF1  79          ld a,c
EEF2  FE 07       cp 07            'BEL (^G)
EEF4  38 04       jr c,EEFA        ...
EEF6  CB 72       bit 6,d
EEF8  20 26       jr nz,EF20       ...
EEFA  B8          cp b
EEFB  C4 A0 EF    call nz,EFA0     ...
EEFE  C3 62 EF    jp EF62          ...

EF01  7B          ld a,e
EF02  B7          or a
EF03  FA 0A EF    jp m,EF0A        ...
EF06  20 DC       jr nz,EEE4       ...
EF08  41          ld b,c
EF09  C9          ret

EF0A  43          ld b,e
EF0B  CD 0E F0    call F00E        ...
EF0E  78          ld a,b
EF0F  B7          or a
EF10  28 F6       jr z,EF08        ...
EF12  93          sub e
EF13  58          ld e,b
EF14  47          ld b,a
EF15  81          add a,c
EF16  83          add a,e
EF17  FA E4 EE    jp m,EEE4        ...
EF1A  CD B4 EF    call EFB4        ...
EF1D  C3 A0 EF    jp EFA0          ...

EF20  3E 06       ld a,06          'ACK (^F)
EF22  32 6E AE    ld (AE6E),a      temp store for char
EF25  18 24       jr EF4B          ...

      @ EED8
EF27  06 80       ld b,80          'F0  0
EF29  CD 25 F0    call F025        ...
EF2C  30 04       jr nc,EF32       ...
EF2E  CD 96 F0    call F096        ...
EF31  AF          xor a
EF32  47          ld b,a
EF33  CC 36 F0    call z,F036      ...
EF36  20 0C       jr nz,EF44       ...
EF38  04          inc b
EF39  3A 6E AE    ld a,(AE6E)      temp store for char
EF3C  B7          or a
EF3D  28 05       jr z,EF44        ...
EF3F  05          dec b
EF40  3C          inc a
EF41  32 6E AE    ld (AE6E),a      temp store for char
EF44  79          ld a,c
EF45  B7          or a
```

```
EF46  28 04      jr z,EF4C       ...
EF48  83         add a,e
EF49  90         sub b
EF4A  5F         ld e,a
EF4B  78         ld a,b
EF4C  F5         push af
EF4D  47          ld b,a
EF4E  CD 8B EF    call EF8B      ...
EF51  F1         pop af
EF52  B8         cp b
EF53  28 0D      jr z,EF62       ...
EF55  1C         inc e
EF56  23         inc hl
EF57  05         dec b
EF58  E5         push hl
EF59  7E          ld a,(hl)
EF5A  FE 2E       cp 2E          '.
EF5C  20 01       jr nz,EF5F     ...
EF5E  23          inc hl
EF5F  36 31       ld (hl),31     '1
EF61  E1         pop hl

      @ EEFE  EF53'
EF62  3E 45      ld a,45         'E
EF64  CD 6F F0   call F06F       ...
EF67  7B         ld a,e
EF68  87         add a,a
EF69  3E 2B      ld a,2B         '+
EF6B  30 05      jr nc,EF72      ...
EF6D  AF         xor a
EF6E  93         sub e
EF6F  5F         ld e,a
EF70  3E 2D      ld a,2D         '-
EF72  CD 6F F0   call F06F       ...
EF75  7B         ld a,e
EF76  0E 2F      ld c,2F         '/
EF78  0C         inc c
EF79  D6 0A      sub 0A          'LF  (^J)
EF7B  30 FB      jr nc,EF78      ...
EF7D  5F         ld e,a
EF7E  79         ld a,c
EF7F  CD 6F F0   call F06F       ...
EF82  7B         ld a,e
EF83  C6 3A      add a,3A        ':
EF85  C3 6F F0   jp F06F         ...

      @ EEDF
EF88  CD B4 EF   call EFB4       ...

      @ EF4E!
EF8B  CD 36 F0   call F036       ...
EF8E  80         add a,b
EF8F  B9         cp c
EF90  30 05      jr nc,EF97      ...
EF92  CD C8 EF   call EFC8       ...
EF95  18 04      jr EF9B         ...

EF97  91         sub c
EF98  C4 EF EF   call nz,EFEF    ...
EF9B  3A 6E AE   ld a,(AE6E)     temp store for char
EF9E  B7         or a
EF9F  C8         ret z
```

```
      @ EEFB! EF1D
EFA0  0E 2E       ld c,2E
EFA2  78          ld a,b

      @ F049!
EFA3  C5          push bc
EFA4  47           ld b,a
EFA5  04           inc b
EFA6  85           add a,l
EFA7  6F           ld l,a
EFA8  8C           adc a,h
EFA9  95           sub l
EFAA  67           ld h,a
EFAB  2B           dec hl
EFAC  79           ld a,c
EFAD  4E           ld c,(hl)
EFAE  77           ld (hl),a
EFAF  05           dec b
EFB0  20 F9        jr nz,EFAB       ...
EFB2  C1          pop bc
EFB3  C9          ret

      @ EF1A! EF88!
EFB4  7B          ld a,e
EFB5  81          add a,c
EFB6  47          ld b,a
EFB7  F0          ret p
EFB8  2F          cpl
EFB9  3C          inc a
EFBA  06 14       ld b,14          'DC4 (^T)
EFBC  B8          cp b
EFBD  30 01       jr nc,EFC0       ...
EFBF  47          ld b,a
EFC0  2B          dec hl
EFC1  36 30       ld (hl),30       '0
EFC3  0C          inc c
EFC4  05          dec b
EFC5  20 F9       jr nz,EFC0       ...
EFC7  C9          ret

      @ EF92!
EFC8  E5          push hl
EFC9  4F           ld c,a
EFCA  85           add a,l
EFCB  6F           ld l,a
EFCC  8C           adc a,h
EFCD  95           sub l
EFCE  67           ld h,a
EFCF  7E           ld a,(hl)
EFD0  36 00        ld (hl),00       'NUL
EFD2  22 70 AE     ld (AE70),hl     number edit buffer index
EFD5  FE 35        cp 35            '5
EFD7  D4 E1 EF     call nc,EFE1     ...
EFDA  E1          pop hl
EFDB  D8          ret c
EFDC  2B          dec hl
EFDD  36 31       ld (hl),31       '1
EFDF  04          inc b
EFE0  C9          ret

      @ EFD7! EFED'
EFE1  79          ld a,c
EFE2  B7          or a
EFE3  C8          ret z
EFE4  2B          dec hl
```

```
EFE5  0D        dec c
EFE6  7E        ld a,(hl)
EFE7  34        inc (hl)
EFE8  FE 39     cp 39           '9
EFEA  D8        ret c
EFEB  36 30     ld (hl),30      '0
EFED  18 F2     jr EFE1         ...

      @ EF98!
EFEF  D5        push de
EFF0  C5         push bc
EFF1  EB          ex de,hl
EFF2  47          ld b,a
EFF3  7B          ld a,e
EFF4  90          sub b
EFF5  6F          ld l,a
EFF6  9F          sbc a,a
EFF7  82          add a,d
EFF8  67          ld h,a
EFF9  E5          push hl
EFFA  0C           inc c
EFFB  18 04        jr F001      ...

EFFD  1A        ld a,(de)
EFFE  13        inc de
EFFF  77        ld (hl),a
F000  23        inc hl
F001  0D        dec c
F002  20 F9     jr nz,EFFD      ...
F004  36 30     ld (hl),30      '0
F006  23        inc hl
F007  05        dec b
F008  20 FA     jr nz,F004      ...
F00A  E1        pop hl
F00B  C1        pop bc
F00C  D1        pop de
F00D  C9        ret

      @ EEEC! EF0B!
F00E  E5        push hl
F00F  2A 70 AE   ld hl,(AE70)   number edit buffer index
F012  2B         dec hl
F013  7E         ld a,(hl)
F014  23         inc hl
F015  FE 30      cp 30          '0
F017  20 05      jr nz,F01E     ...
F019  2B         dec hl
F01A  0D         dec c
F01B  04         inc b
F01C  20 F4      jr nz,F012     ...
F01E  36 00      ld (hl),00     'NUL
F020  22 70 AE   ld (AE70),hl   number edit buffer index
F023  E1        pop hl
F024  C9        ret

      @ EF29! F055!
F025  CD 9B F0  call F09B       ...
F028  9F        sbc a,a
F029  3C        inc a
F02A  47        ld b,a
F02B  7A        ld a,d
F02C  E6 04     and 04          'EOT (^D)
F02E  28 01     jr z,F031       ...
F030  04        inc b
F031  3A 6F AE  ld a,(AE6F)     ...
```

```
F034  90          sub b
F035  C9          ret

      @ EF33! EF8B!
F036  3A 6E AE    ld a,(AE6E)      temp store for char
F039  B7          or a
F03A  C8          ret z
F03B  3D          dec a
F03C  C9          ret

      @ EEBA!
F03D  7A          ld a,d
F03E  E6 02       and 02           'STX (^B)
F040  C8          ret z
F041  78          ld a,b
F042  D6 03       sub 03           'ETX (^C)
F044  D8          ret c
F045  C8          ret z
F046  F5          push af
F047  0E 2C        ld c,2C         ',
F049  CD A3 EF     call EFA3       ...
F04C  04           inc b
F04D  F1          pop af
F04E  18 F2       jr F042          ...

      @ EEC1!
F050  7A          ld a,d
F051  87          add a,a
F052  30 07       jr nc,F05B       ...
F054  C5          push bc
F055  CD 25 F0     call F025       ...
F058  C1          pop bc
F059  D8          ret c
F05A  C8          ret z
F05B  3E 30       ld a,30          '0
F05D  18 06       jr F065          ...

      @ EEC4!
F05F  7A          ld a,d
F060  E6 04       and 04           'EOT (^D)
F062  C8          ret z
F063  3E 24       ld a,24          '$
F065  1C          inc e
F066  2B          dec hl
F067  77          ld (hl),a
F068  C9          ret

      @ EEC7!
F069  CD 9B F0    call F09B        ...
F06C  C8          ret z
F06D  30 F6       jr nc,F065       ...

      @ EF64! EF72! EF7F! EF85
F06F  E5          push hl
F070  2A 70 AE     ld hl,(AE70)    number edit buffer index
F073  77           ld (hl),a
F074  23           inc hl
F075  36 00        ld (hl),00      'NUL
F077  22 70 AE     ld (AE70),hl    number edit buffer index
F07A  E1          pop hl
F07B  C9          ret
```

```
        @ EECA!
F07C  7A           ld a,d
F07D  B7           or a
F07E  F0           ret p
F07F  3A 6F AE     ld a,(AE6F)        ...
F082  93           sub e
F083  C8           ret z
F084  38 10        jr c,F096          ...
F086  47           ld b,a
F087  7A           ld a,d
F088  E6 20        and 20             'SPACE
F08A  3E 2A        ld a,2A            '*
F08C  20 02        jr nz,F090         ...
F08E  3E 20        ld a,20            'SPACE
F090  2B           dec hl
F091  77           ld (hl),a
F092  05           dec b
F093  20 FB        jr nz,F090         ...
F095  C9           ret

        @ EF2E! F084'
F096  7A           ld a,d
F097  F6 01        or 01              'SOH (^A)
F099  57           ld d,a
F09A  C9           ret

        @ F025! F069!
F09B  78           ld a,b
F09C  06 2D        ld b,2D            '-
F09E  87           add a,a
F09F  38 0F        jr c,F0B0          ...
F0A1  7A           ld a,d
F0A2  E6 98        and 98             'F24
F0A4  EE 80        xor 80             'F0  0
F0A6  37           scf
F0A7  C8           ret z
F0A8  06 2B        ld b,2B            '+
F0AA  E6 08        and 08             'BS  (^H)
F0AC  20 02        jr nz,F0B0         ...
F0AE  06 20        ld b,20            'SPACE
F0B0  7A           ld a,d
F0B1  F6 EF        or EF              'BREAK mark
F0B3  C6 10        add a,10           'DLE (^P)
F0B5  78           ld a,b
F0B6  C9           ret

        @ EEB3!
F0B7  E5           push hl
F0B8  EB            ex de,hl
F0B9  CD DD F0      call F0DD         ...
F0BC  E1           pop hl
F0BD  78           ld a,b
F0BE  87           add a,a
F0BF  4F           ld c,a
F0C0  C8           ret z
F0C1  1A           ld a,(de)
F0C2  E6 0F        and 0F             'SI  (^O)
F0C4  C6 30        add a,30           '0
F0C6  2B           dec hl
F0C7  77           ld (hl),a
F0C8  1A           ld a,(de)
F0C9  E6 F0        and F0             'CURSOR up
F0CB  1F           rra
F0CC  1F           rra
F0CD  1F           rra
```

```
F0CE  1F          rra
F0CF  C6 30       add a,30        '0
F0D1  2B          dec hl
F0D2  77          ld (hl),a
F0D3  13          inc de
F0D4  05          dec b
F0D5  20 EA       jr nz,F0C1      ...
F0D7  FE 30       cp 30           '0
F0D9  C0          ret nz
F0DA  0D          dec c
F0DB  23          inc hl
F0DC  C9          ret

      @ F0B9!
F0DD  11 46 AE    ld de,AE46      number edit buffer (1)
F0E0  AF          xor a
F0E1  47          ld b,a
F0E2  B6          or (hl)
F0E3  2B          dec hl
F0E4  20 04       jr nz,F0EA      ...
F0E6  0D          dec c
F0E7  20 F9       jr nz,F0E2      ...
F0E9  C9          ret

F0EA  37          scf
F0EB  8F          adc a,a
F0EC  30 FD       jr nc,F0EB      ...
F0EE  EB          ex de,hl
F0EF  D5          push de
F0F0  57           ld d,a
F0F1  18 11        jr F104        ...

F0F3  1A          ld a,(de)
F0F4  1B          dec de
F0F5  D5          push de
F0F6  37           scf
F0F7  8F           adc a,a
F0F8  57           ld d,a
F0F9  58           ld e,b
F0FA  7E           ld a,(hl)
F0FB  8F           adc a,a
F0FC  27           daa
F0FD  77           ld (hl),a
F0FE  23           inc hl
F0FF  1D           dec e
F100  20 F8        jr nz,F0FA     ...
F102  30 03        jr nc,F107     ...
F104  04           inc b
F105  36 01        ld (hl),01     'SOH (^A)
F107  21 46 AE     ld hl,AE46     number edit buffer (1)
F10A  7A           ld a,d
F10B  87           add a,a
F10C  20 EA        jr nz,F0F8     ...
F10E  D1          pop de
F10F  0D          dec c
F110  20 E1       jr nz,F0F3      ...
F112  EB          ex de,hl
F113  C9          ret

----- edit value in binary representation
      @ E2A7! F8BE!
F114  11 01 01    ld de,0101      d= # of bits per digit, e= base 2 (-1)
F117  18 03       jr F11C
```

```
----- edit value in HEX representation
      @ E2B2! F8C8!
F119  11 0F 04    ld de,040F        d= # of bits per digit, e= base 16 (-1)
F11C  D5          push de
F11D  79           ld a,c           = VARTYPE
F11E  CD 4B FF     call FF4B        set VARTYPE <a>, copy VARIABLE (hl) to FAC
F121  E3           ex (sp),hl
F122  E5           push hl
F123  C5            push bc
F124  CD C2 FE       call FEC2      function: UNT(<address expression>)
F127  11 57 AE       ld de,AE57     ...
F12A  AF             xor a          =0
F12B  12             ld (de),a      mark end of text
F12C  F1            pop af          a= min # of digits to produce
F12D  C1           pop bc           b= # of bits per digit, c= base

----- produce a digit
F12E  D6 01       sub 01            'SOH (^A)
F130  CE 00       adc a,00          'NUL
F132  F5          push af
F133  7D           ld a,l           get LSB
F134  A1           and c            mask out unwanted bits
F135  F6 F0        or F0            set upper nibble
F137  27           daa              decimal adjust both nibbles
F138  C6 A0        add a,A0
F13A  CE 40        adc a,40         change to ascii
F13C  1B           dec de
F13D  12           ld (de),a        store digit on NUMBER EDIT STACK
F13E  7D           ld a,l
F13F  B1           or c
F140  A9           xor c
F141  6F           ld l,a
F142  B4           or h
F143  28 0E        jr z,F153        all bits done
F145  C5           push bc
F146  7C            ld a,h          shift <hl> right <b> times
F147  1F            rra
F148  67            ld h,a
F149  7D            ld a,l
F14A  1F            rra
F14B  6F            ld l,a
F14C  05            dec b           decrement bit count
F14D  20 F7         jr nz,F146      shift again
F14F  C1           pop bc
F150  F1          pop af
F151  18 DB       jr F12E           produce a digit

F153  F1          pop af
F154  20 D8       jr nz,F12E        digit count not done, add a zero
F156  E1          pop hl            points to variable within basic text
F157  C9          ret

----- function: PEEK (<address>)
      @ D1D2
F158  CD C2 FE    call FEC2         function: UNT(<address expression>)
F15B  E7          rst 4             ld a,(hl) from RAM
F15C  C3 0A FF    jp FF0A           set FAC to <a> and mark integer

----- command: POKE <address>,<byte value>
      @ DE7D
F15F  CD 91 CE    call CE91         get unsigned-integer VAL(expr) in <de>
F162  D5          push de           save address argument
F163  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
F166  2C           ',
F167  CD 67 CE     call CE67        get byte VAL(expression) in <de>
```

```
F16A  D1          pop de          get address
F16B  12          ld (de),a       and store
F16C  C9          ret

----- function: INP (<I/O address>)
      @ D1C4
F16D  CD 8D FE    call FE8D       function: CINT(<num expression>) in <hl>
F170  44          ld b,h
F171  4D          ld c,l          bc=hl
F172  ED 78       in a,(c)        read I/O port
F174  C3 0A FF    jp FF0A         set FAC to <a> and mark integer

----- command: OUT <I/O address>,<byte value>
      @ DE73
F177  CD 94 F1    call F194       get int VAL in <bc>, next byte VAL in <a>
F17A  ED 79       out (c),a
F17C  C9          ret

----- command: WAIT <I/O address>,<AND mask>[,<XOR mask>]
      @ DEA9
F17D  CD 94 F1    call F194       get int VAL in <bc>, next byte VAL in <a>
F180  57          ld d,a
F181  1E 00       ld e,00         default XOR mask
F183  28 08       jr z,F18D       wait for condition
F185  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
F188  2C          ',
F189  CD 67 CE    call CE67       get byte VAL(expression) in <de>
F18C  5F          ld e,a

----- wait for condition
F18D  ED 78       in a,(c)        read I/O port
F18F  AB          xor e
F190  A2          and d
F191  28 FA       jr z,F18D       wait for condition
F193  C9          ret

----- get int VAL in <bc>, next byte VAL in <a>
      @ F177! F17D!
F194  CD 91 CE    call CE91       get unsigned-integer VAL(expr) in <de>
F197  42          ld b,d          bc=de
F198  4B          ld c,e
F199  CD 37 DD    call DD37       CHRNEXT <a>, nz=Error; CHRGET
F19C  2C          ',
F19D  C3 67 CE    jp CE67         get byte VAL(expression) in <de>

----- perform an EXTERNAL CALL
      @ D651
F1A0  23          inc hl
F1A1  7E          ld a,(hl)
F1A2  B7          or a
F1A3  20 10       jr nz,F1B5      Error: Unknown command
F1A5  23          inc hl
F1A6  E5          push hl
F1A7  CD D4 BC     call BCD4      KL FIND COMMAND (hl) in RSX or back ROM, =<c
F1AA  EB           ex de,hl
F1AB  E1          pop hl
F1AC  30 07       jr nc,F1B5      Error: Unknown command

----- skip over command
F1AE  7E          ld a,(hl)
F1AF  23          inc hl
F1B0  17          rla
F1B1  30 FB       jr nc,F1AE      skip over command
F1B3  18 0A       jr F1BF         copy arg's to stack, call routine in selecte
```

```
----- Error: Unknown command
F1B5  1E 1C        ld e,1C           Unknown command
F1B7  C3 94 CA     jp CA94           perform ERROR <e> routine

----- command: CALL <RAM address>[,<list of<argument>>]
      @ DE07
F1BA  CD 91 CE     call CE91         get unsigned-integer VAL(expr) in <de>
F1BD  0E FF        ld c,FF           disable ROMs on CALL

----- copy arg's to stack, call routine in selected ROM
F1BF  ED 53 72 AE ld (AE72),de       address of CALLed routine
F1C3  79           ld a,c
F1C4  32 74 AE     ld (AE74),a       ROM selection on CALL
F1C7  ED 73 77 AE ld (AE77),sp       save SP on CALL
F1CB  06 20        ld b,20           count for up to 32. arguments

----- get all the arguments
F1CD  CD 55 DD     call DD55         CHRBACK comma?; if=:CHRGET <a>, scf
F1D0  30 06        jr nc,F1D8        skip if no more arguments
F1D2  CD 91 CE     call CE91         get unsigned-integer VAL(expr) in <de>
F1D5  D5           push de           copy arguments onto stack
F1D6  10 F5         djnz F1CD        get all the arguments
F1D8  CD 4A DD      call DD4A        CHRGOT <a>; end of statement? else syntax er
F1DB  22 75 AE      ld (AE75),hl     save HL on CALL
F1DE  3E 20         ld a,20          max # of arguments
F1E0  90            sub b            - unused = # of arg's in <a>
F1E1  DD 21 00 00  ld ix,0000        clear ix
F1E5  DD 39         add ix,sp        ix=index to find arg's on stack
F1E7  DF 72 AE      rst 3,AE72       call routine

----- restore SP and HL after return from CALL
F1EA  ED 7B 77 AE ld sp,(AE77)       save SP on CALL
F1EE  2A 75 AE     ld hl,(AE75)      save HL on CALL
F1F1  C9           ret

----- set default ZONE to 13.
      @ C171!
F1F2  3E 0D        ld a,0D           'CR  (^M)
F1F4  18 03        jr F1F9           set ZONE

----- command: ZONE <byte value>
      @ DEB5
F1F6  CD 6D CE     call CE6D         <a>=next VAL, 0=error
F1F9  32 79 AE     ld (AE79),a       ZONE for TAB
F1FC  C9           ret

----- command: PRINT [#<device>,][<list of<variable>>]
      @ DE7F
F1FD  CD C6 C1     call C1C6         if '# get chan; default=0; set output channe
F200  F5           push af
F201  CD 08 F2      call F208        do it here
F204  F1           pop af
F205  C3 A2 C1     jp C1A2           set output channel to <a>, a= old channel

----- do it here
      @ F201!
F208  CD 51 DD     call DD51         CHRGOT <a>; end of statement? =carry
F20B  DA 4E C3     jp c,C34E         output 'LF to channel

----- print a variable
F20E  FE ED        cp ED             [USING]
F210  CA C4 F2     jp z,F2C4         command: ,USING <format>[<separator>]
F213  EB           ex de,hl
F214  21 24 F2     ld hl,F224        TOKEN: COMMA SPC TAB SEMICOLON
F217  CD 93 FF     call FF93         search <a> in table(hl); hl=address
```

```
F21A  EB         ex de,hl
F21B  CD FB FF   call FFFB        jp(de)
F21E  CD 51 DD   call DD51        CHRGOT <a>; end of statement? =carry
F221  30 EB      jr nc,F20E       print a variable
F223  C9         ret

----- TOKEN: COMMA SPC TAB SEMICOLON
      @ F214:
F224  05 33 F2   F233 =5.         continue here if none of these
F227  2C 5C F2   F25C ',          function: ',' (TAB)
F22A  E5 77 F2   F277 [SPC]       function: SPC(<spaces>)
F22D  EA 80 F2   F280 [TAB]       function: TAB(<position>)
F230  3B 3F DD   DD3F ';          CHRGET <a>, skip blank, cp 01

----- continue here if none of these
      @ F224:
F233  CD FB CE   call CEFB        evaluate (expression), CHRGET, cp 01
F236  F5         push af
F237  E5          push hl
F238  CD 45 FF     call FF45      get VARTYPE <a>; cp string
F23B  28 0C        jr z,F249      it is a string
F23D  CD 9D EE     call EE9D      ...
F240  CD DC F7     call F7DC      test len of string; copy temp to stack
F243  36 20        ld (hl),20     'SPACE
F245  2A C2 B0     ld hl,(B0C2)   Floating point ACU, FAC
F248  34           inc (hl)

----- it is a string
F249  2A C2 B0     ld hl,(B0C2)   Floating point ACU, FAC
F24C  7E           ld a,(hl)      hl points to string, get a char
F24D  CD B9 C2     call C2B9      check if char fits into this line
F250  D4 4E C3     call nc,C34E   output 'LF to channel
F253  CD 28 F8     call F828      print this line
F256  E1          pop hl
F257  F1         pop af           append 'LF if no argument nor ';
F258  CC 4E C3   call z,C34E      output 'LF to channel
F25B  C9         ret

----- function: ',' (TAB)
      @ F227:
F25C  CD 3F DD   call DD3F        CHRGET <a>, skip blank, cp 01
F25F  3A 79 AE   ld a,(AE79)      ZONE for TAB
F262  4F         ld c,a
F263  CD 90 C2   call C290        get current print POS of <device>
F266  3D         dec a
F267  91         sub c
F268  30 FD      jr nc,F267       find remainder
F26A  2F         cpl
F26B  3C         inc a
F26C  47         ld b,a
F26D  81         add a,c
F26E  CD B9 C2   call C2B9        check if char fits into this line
F271  D2 4E C3   jp nc,C34E       output 'LF to channel
F274  78         ld a,b
F275  18 1E      jr F295          print <a> spaces to current channel

----- function: SPC(<spaces>)
      @ F22A:
F277  CD A0 F2   call F2A0        get integer VAL(expression)
F27A  CD AF F2   call F2AF        ...
F27D  7B         ld a,e
F27E  18 15      jr F295          print <a> spaces to current channel
```

```
----- function: TAB(<position>)
      @ F22D:
F280  CD A0 F2    call F2A0         get integer VAL(expression)
F283  1B          dec de
F284  CD AF F2    call F2AF         ...
F287  CD 90 C2    call C290         get current print POS of <device>
F28A  2F          cpl
F28B  3C          inc a
F28C  1C          inc e
F28D  83          add a,e
F28E  38 05       jr c,F295         print <a> spaces to current channel
F290  CD 4E C3    call C34E         output 'LF to channel
F293  1D          dec e
F294  7B          ld a,e

----- print <a> spaces to current channel
      @ F275' F27E' F28E' F39C!
F295  47          ld b,a
F296  04          inc b
F297  05          dec b
F298  C8          ret z
F299  3E 20       ld a,20           'SPACE
F29B  CD 56 C3    call C356         output char <a> to channel
F29E  18 F7       jr F297           next

----- get integer VAL(expression)
      @ F277! F280!
F2A0  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
F2A3  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
F2A6  28          '(
F2A7  CD 86 CE    call CE86         get integer VAL(expression) in <de>
F2AA  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
F2AD  29          ')
F2AE  C9          ret

      @ F27A! F284!
F2AF  7A          ld a,d
F2B0  17          rla
F2B1  30 03       jr nc,F2B6        ...
F2B3  11 00 00    ld de,0000
F2B6  CD 9F C2    call C29F         get line width of <device>
F2B9  D0          ret nc
F2BA  E5          push hl
F2BB  EB           ex de,hl
F2BC  5F           ld e,a
F2BD  16 00        ld d,00          'NUL
F2BF  CD C1 BD     call BDC1        INT ARITH, DVDu <hl>=<hl>/<de>; <de>=remaind
F2C2  E1          pop hl
F2C3  C9          ret

----- command: ,USING <format>[<separator>]
      @ F210
F2C4  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
F2C7  CD A5 CE    call CEA5         evaluate (string expression)
F2CA  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
F2CD  3B          ';
F2CE  E5          push hl
F2CF  2A C2 B0     ld hl,(B0C2)     Floating point ACU, FAC
F2D2  7E           ld a,(hl)        len of mask string
F2D3  B7           or a             =0?  =error
F2D4  28 75        jr z,F34B        Error: Improper argument
F2D6  E3           ex (sp),hl
F2D7  CD FB CE     call CEFB        evaluate (expression), CHRGET, cp 01
F2DA  AF           xor a
F2DB  32 7A AE     ld (AE7A),a      flag for PRINT USING
```

```
        @ F306' F30B'
F2DE  D1          pop de
F2DF  D5          push de
F2E0  EB           ex de,hl
F2E1  46           ld b,(hl)
F2E2  23           inc hl
F2E3  7E           ld a,(hl)
F2E4  23           inc hl
F2E5  66           ld h,(hl)
F2E6  6F           ld l,a
F2E7  EB           ex de,hl
F2E8  CD 24 F3     call F324      ...
F2EB  30 5E        jr nc,F34B     Error: Improper argument
F2ED  CD 51 DD     call DD51      CHRGOT <a>; end of statement? =carry
F2F0  38 1D        jr c,F30F      ...
F2F2  FE 3B        cp 3B          ';
F2F4  28 04        jr z,F2FA      ...
F2F6  FE 2C        cp 2C          ',
F2F8  20 4C        jr nz,F346     Error: Syntax error
F2FA  CD 3F DD     call DD3F      CHRGET <a>, skip blank, cp 01
F2FD  28 10        jr z,F30F      ...
F2FF  D5           push de
F300  CD FB CE      call CEFB     evaluate (expression), CHRGET, cp 01
F303  D1           pop de
F304  78           ld a,b
F305  B7           or a
F306  28 D6        jr z,F2DE      ...
F308  CD 24 F3     call F324      ...
F30B  30 D1        jr nc,F2DE     ...
F30D  18 DE        jr F2ED        ...

        @ F2F0' F2FD'
F30F  F5           push af
F310  3E FF         ld a,FF       'IGNORE
F312  32 7A AE      ld (AE7A),a   flag for PRINT USING
F315  78            ld a,b
F316  B7            or a
F317  C4 24 F3      call nz,F324  ...
F31A  F1           pop af
F31B  DC 4E C3     call c,C34E    output 'LF to channel
F31E  E3           ex (sp),hl
F31F  CD E8 FB     call FBE8      try to release string at low end of string s
F322  E1          pop hl
F323  C9          ret

        @ F2E8! F308! F317!
F324  E5          push hl
F325  1A           ld a,(de)      no leading sign
F326  FE 5F        cp 5F          '_
F328  20 09        jr nz,F333     ...
F32A  78           ld a,b
F32B  FE 02        cp 02          'STX (^B)
F32D  38 0C        jr c,F33B      ...
F32F  13           inc de
F330  05           dec b
F331  18 08        jr F33B        ...

F333  CD 50 F3     call F350      ...
F336  D4 A3 F3     call nc,F3A3   ...
F339  38 09        jr c,F344      ...
F33B  1A           ld a,(de)
F33C  CD 56 C3     call C356      output char <a> to channel
F33F  13           inc de
F340  05           dec b
F341  20 E2        jr nz,F325     ...
```

```
F343  B7          or a
F344  E1         pop hl
F345  C9         ret

----- Error: Syntax error
F346  1E 02       ld e,02          Syntax error
F348  C3 94 CA    jp CA94          perform ERROR <e> routine

----- Error: Improper argument
F34B  1E 05       ld e,05          Improper argument
F34D  C3 94 CA    jp CA94          perform ERROR <e> routine

      @ F333!
F350  1A         ld a,(de)
F351  FE 21      cp 21             '!
F353  0E 01      ld c,01
F355  28 21      jr z,F378         ...
F357  FE 26      cp 26             '&
F359  0E 00      ld c,00
F35B  28 1B      jr z,F378         ...
F35D  EE 5C      xor 5C            '\
F35F  C0         ret nz
F360  C5         push bc
F361  D5          push de
F362  0E 02        ld c,02
F364  13           inc de
F365  05           dec b
F366  28 0A        jr z,F372       ...
F368  1A           ld a,(de)
F369  FE 5C        cp 5C           '\
F36B  28 09        jr z,F376       ...
F36D  0C           inc c
F36E  FE 20        cp 20           'SPACE
F370  28 F2        jr z,F364       ...
F372  D1          pop de
F373  C1         pop bc
F374  B7         or a
F375  C9         ret

F376  F1          pop af
F377  F1         pop af
F378  13         inc de
F379  05         dec b
F37A  C5         push bc
F37B  D5          push de
F37C  3A 7A AE     ld a,(AE7A)     flag for PRINT USING
F37F  B7           or a
F380  20 1D        jr nz,F39F      ...
F382  CD 3C FF     call FF3C       test VARTYPE for string, else error
F385  79           ld a,c
F386  B7           or a
F387  F5           push af
F388  41            ld b,c
F389  0E 00         ld c,00
F38B  2A C2 B0      ld hl,(B0C2)   Floating point ACU, FAC
F38E  EB            ex de,hl
F38F  C4 71 F9      call nz,F971   ...
F392  CD 28 F8      call F828      print this line
F395  F1           pop af
F396  28 07        jr z,F39F       ...
F398  2A C2 B0     ld hl,(B0C2)    Floating point ACU, FAC
F39B  96           sub (hl)
F39C  CD 95 F2     call F295       print <a> spaces to current channel
F39F  D1          pop de
F3A0  C1         pop bc
```

```
F3A1  37          scf
F3A2  C9          ret

      @ F336!
F3A3  CD BA F3    call F3BA         ...
F3A6  D0          ret nc
F3A7  3A 7A AE    ld a,(AE7A)       flag for PRINT USING
F3AA  B7          or a
F3AB  20 0B       jr nz,F3B8        ...
F3AD  C5          push bc
F3AE  D5           push de
F3AF  79            ld a,c
F3B0  CD 9F EE      call EE9F       ...
F3B3  CD 41 C3      call C341       output text (hl) to channel
F3B6  D1           pop de
F3B7  C1          pop bc
F3B8  37          scf
F3B9  C9          ret

      @ F3A3! F90C!
F3BA  C5          push bc
F3BB  D5           push de
F3BC  0E 80         ld c,80         'F0  0
F3BE  26 00         ld h,00         'NUL
F3C0  1A            ld a,(de)
F3C1  FE 2B         cp 2B           '+
F3C3  20 07         jr nz,F3CC      ...
F3C5  13            inc de
F3C6  05            dec b
F3C7  28 23         jr z,F3EC       ...
F3C9  24            inc h
F3CA  0E 88         ld c,88         'F8  8
F3CC  1A            ld a,(de)
F3CD  FE 2E         cp 2E           '.
F3CF  28 1F         jr z,F3F0       ...
F3D1  FE 23         cp 23           '#
F3D3  28 39         jr z,F40E       ...
F3D5  13            inc de
F3D6  05            dec b
F3D7  28 13         jr z,F3EC       ...
F3D9  EB            ex de,hl
F3DA  BE            cp (hl)
F3DB  EB            ex de,hl
F3DC  20 0E         jr nz,F3EC      ...
F3DE  24            inc h
F3DF  24            inc h
F3E0  2E 04         ld 1,04
F3E2  FE 24         cp 24           '$
F3E4  28 23         jr z,F409       ...
F3E6  2E 20         ld 1,20         'SPACE
F3E8  FE 2A         cp 2A           '*
F3EA  28 11         jr z,F3FD       ...
F3EC  D1           pop de
F3ED  C1          pop bc
F3EE  B7          or a
F3EF  C9          ret

F3F0  13            inc de
F3F1  05            dec b
F3F2  28 F8         jr z,F3EC       ...
F3F4  1A            ld a,(de)
F3F5  FE 23         cp 23           '#
F3F7  20 F3         jr nz,F3EC      ...
F3F9  1B            dec de
F3FA  04            inc b
```

```
F3FB  18 11        jr F40E          ...

F3FD  13           inc de
F3FE  05           dec b
F3FF  28 0A        jr z,F40B        ...
F401  1A           ld a,(de)
F402  FE 24        cp 24            '$
F404  20 05        jr nz,F40B       ...
F406  24           inc h
F407  2E 24        ld 1,24          '$
F409  13           inc de
F40A  05           dec b
F40B  79           ld a,c
F40C  B5           or 1
F40D  4F           ld c,a
F40E  F1           pop af
F40F  F1           pop af
F410  CD 1B F4     call F41B        ...
F413  7C           ld a,h
F414  85           add a,1
F415  FE 15        cp 15
F417  D2 4B F3     jp nc,F34B       Error: Improper argument
F41A  C9           ret

      @ F410!
F41B  2E 00        ld 1,00          'NUL
F41D  04           inc b
F41E  05           dec b
F41F  C8           ret z
F420  1A           ld a,(de)
F421  FE 2E        cp 2E            '.
F423  28 14        jr z,F439        ...
F425  FE 2C        cp 2C            ',
F427  28 0A        jr z,F433        ...
F429  FE 23        cp 23            '#
F42B  20 15        jr nz,F442       ...
F42D  24           inc h
F42E  13           inc de
F42F  05           dec b
F430  20 EE        jr nz,F420       ...
F432  C9           ret

F433  79           ld a,c
F434  F6 02        or 02            'STX (^B)
F436  4F           ld c,a
F437  18 F4        jr F42D          ...

F439  2C           inc 1
F43A  13           inc de
F43B  05           dec b
F43C  C8           ret z
F43D  1A           ld a,(de)
F43E  FE 23        cp 23            '#
F440  28 F7        jr z,F439        ...
F442  EB           ex de,hl
F443  E5           push hl
F444  FE 5E        cp 5E            '^
F446  20 18        jr nz,F460       ...
F448  23           inc hl
F449  BE           cp (hl)
F44A  20 14        jr nz,F460       ...
F44C  23           inc hl
F44D  BE           cp (hl)
F44E  20 10        jr nz,F460       ...
F450  23           inc hl
```

```
F451  BE            cp (hl)
F452  20 0C         jr nz,F460        ...
F454  23            inc hl
F455  78            ld a,b
F456  D6 04         sub 04            'EOT (^D)
F458  38 06         jr c,F460         ...
F45A  47            ld b,a
F45B  E3            ex (sp),hl
F45C  79            ld a,c
F45D  F6 40         or 40             '@
F45F  4F            ld c,a
F460  E1           pop hl
F461  EB           ex de,hl
F462  78           ld a,b
F463  B7           or a
F464  C8           ret z
F465  79           ld a,c
F466  E6 08        and 08             'BS  (^H)
F468  C0           ret nz
F469  1A           ld a,(de)
F46A  FE 2D        cp 2D              '-
F46C  3E 10        ld a,10            'DLE (^P)
F46E  28 06        jr z,F476          ...
F470  1A           ld a,(de)
F471  FE 2B        cp 2B              '+
F473  C0           ret nz
F474  3E 18        ld a,18            'CAN (^X)
F476  B1           or c
F477  4F           ld c,a
F478  13           inc de
F479  05           dec b
F47A  C9           ret

----- command: WRITE [#<device>,][<list of<variable>>]
      @ DEB3
F47B  CD C6 C1     call C1C6          if '# get chan; default=0; set output channe
F47E  F5           push af
F47F  CD 51 DD      call DD51         CHRGOT <a>; end of statement? =carry
F482  38 39         jr c,F4BD         output 'LF and restore old channel
F484  CD FB CE      call CEFB         evaluate (expression), CHRGET, cp 01
F487  F5            push af
F488  E5             push hl
F489  CD 45 FF        call FF45       get VARTYPE <a>; cp string
F48C  28 0B           jr z,F499       output text in '"
F48E  CD 8F EE        call EE8F       edit FAC onto the NUMBER EDIT BUFFER
F491  CD DC F7        call F7DC       test len of string; copy temp to stack
F494  CD 28 F8        call F828       print this line
F497  18 0D           jr F4A6

----- output text in '"
      @ F48C'
F499  3E 22           ld a,22         '"
F49B  CD 56 C3        call C356       output char <a> to channel
F49E  CD 28 F8        call F828       print this line
F4A1  3E 22           ld a,22         '"
F4A3  CD 56 C3        call C356       output char <a> to channel
F4A6  E1             pop hl
F4A7  F1            pop af
F4A8  28 13         jr z,F4BD         output 'LF and restore old channel
F4AA  FE 3B         cp 3B             ';
F4AC  28 05         jr z,F4B3         output ',' and take next expression
F4AE  FE 2C         cp 2C             ',
F4B0  C2 46 F3      jp nz,F346        Error: Syntax error
```

```
----- output ',' and take next expression
F4B3  CD 3F DD     call DD3F        CHRGET <a>, skip blank, cp 01
F4B6  3E 2C        ld a,2C          ',
F4B8  CD 56 C3     call C356        output char <a> to channel
F4BB  18 C7        jr F484          next expression

----- output 'LF and restore old channel
      @ F482' F4A8'
F4BD  CD 4E C3     call C34E        output 'LF to channel
F4C0  F1           pop af
F4C1  C3 A2 C1     jp C1A2          set output channel to <a>, a= old channel

----- init all Basic pointers
      @ C00C!
F4C4  01 00 AC     ld bc,AC00       default HIMEM
F4C7  CD BE FF     call FFBE        test HL=BC? (try hl-bc)
F4CA  D0           ret nc
F4CB  22 7B AE     ld (AE7B),hl     himem for Basic pointer
F4CE  22 8F B0     ld (B08F),hl     upper bound for string space pointer
F4D1  22 7D AE     ld (AE7D),hl     himem for SYMBOL AFTER pointer
F4D4  EB           ex de,hl
F4D5  22 7F AE     ld (AE7F),hl     low memory boundary pointer
F4D8  01 2F 01     ld bc,012F       working space below Basic program
F4DB  09           add hl,bc
F4DC  D8           ret c
F4DD  22 81 AE     ld (AE81),hl     start of BASIC program -1 pointer
F4E0  EB           ex de,hl
F4E1  23           inc hl
F4E2  B7           or a
F4E3  ED 52        sbc hl,de
F4E5  D8           ret c
F4E6  7C           ld a,h
F4E7  FE 04        cp 04            attempt to access system space
F4E9  D8           ret c            forces a system reset
F4EA  AF           xor a
F4EB  32 91 B0     ld (B091),a      tape buffer flag
F4EE  C9           ret

----- command: MEMORY <address>
      @ DE55
F4EF  CD 3E FC     call FC3E        GARBAGE COLLECT
F4F2  CD 91 CE     call CE91        get unsigned-integer VAL(expr) in <de>
F4F5  E5           push hl
F4F6  CD 50 F7     call F750        ...
F4F9  CD 75 F6     call F675        release buffer
F4FC  22 7B AE     ld (AE7B),hl     himem for Basic pointer
F4FF  E1           pop hl
F500  C9           ret

----- enough space in memory for <bc>?
      @ EA02!
F501  D5           push de
F502  2A 7F AE     ld hl,(AE7F)     low memory boundary pointer
F505  EB           ex de,hl
F506  2A 7B AE     ld hl,(AE7B)     himem for Basic pointer
F509  CD CF FF     call FFCF        HL=HL-DE
F50C  E3           ex (sp),hl
F50D  CD CF FF     call FFCF        HL=HL-DE
F510  D1           pop de
F511  13           inc de
F512  CD B8 FF     call FFB8        test HL=DE? (try hl-de)
F515  38 03        jr c,F51A        Error: Memory full
F517  2B           dec hl
F518  09           add hl,bc
F519  D0           ret nc
```

```
F51A  C3 3E F7    jp F73E           Error: Memory full

----- <bc>= space occupied by strings
      @ F75C! F78D! F7A0!
F51D  D5          push de
F51E  E5           push hl
F51F  2A 8D B0      ld hl,(B08D)    low end of used string space pointer
F522  EB            ex de,hl
F523  2A 8F B0      ld hl,(B08F)    upper bound for string space pointer
F526  CD DA FF      call FFDA       BC=HL-DE
F529  E1           pop hl
F52A  D1          pop de
F52B  C9          ret

----- shift all Basic pointers up by <bc>
      @ E6F3! E725
F52C  2A 83 AE    ld hl,(AE83)      end of BASIC program pointer
F52F  09          add hl,bc
F530  22 83 AE    ld (AE83),hl      end of BASIC program pointer
F533  2A 85 AE    ld hl,(AE85)      start of VAR table pointer
F536  09          add hl,bc
F537  22 85 AE    ld (AE85),hl      start of VAR table pointer

----- shift DIM'd VAR pointers up by <bc>
      @ D758!
F53A  2A 87 AE    ld hl,(AE87)      start of DIM'd VAR table pointer
F53D  09          add hl,bc
F53E  22 87 AE    ld (AE87),hl      start of DIM'd VAR table pointer
F541  2A 89 AE    ld hl,(AE89)      upper end of DIM'd variables pointer
F544  09          add hl,bc
F545  22 89 AE    ld (AE89),hl      upper end of DIM'd variables pointer
F548  C9          ret

      @ EA7A!
F549  2A 85 AE    ld hl,(AE85)      start of VAR table pointer
F54C  EB          ex de,hl
F54D  2A 87 AE    ld hl,(AE87)      start of DIM'd VAR table pointer
F550  CD CF FF    call FFCF         HL=HL-DE
F553  E5          push hl
F554  2A 89 AE     ld hl,(AE89)     upper end of DIM'd variables pointer
F557  CD DA FF     call FFDA        BC=HL-DE
F55A  C5           push bc
F55B  2A 8D B0      ld hl,(B08D)    low end of used string space pointer
F55E  EB            ex de,hl
F55F  2A 89 AE      ld hl,(AE89)    upper end of DIM'd variables pointer
F562  2B            dec hl
F563  78            ld a,b
F564  B1            or c
F565  C4 F5 FF      call nz,FFF5    lddr
F568  EB            ex de,hl
F569  22 8D B0      ld (B08D),hl    low end of used string space pointer
F56C  C1           pop bc
F56D  D1          pop de
F56E  C3 B1 D5    jp D5B1           reset all VARIABLE pointers to basic start

      @ EA97!
F571  2A 83 AE    ld hl,(AE83)      end of BASIC program pointer
F574  22 85 AE    ld (AE85),hl      start of VAR table pointer
F577  EB          ex de,hl
F578  19          add hl,de
F579  22 87 AE    ld (AE87),hl      start of DIM'd VAR table pointer
F57C  2A 8D B0    ld hl,(B08D)      low end of used string space pointer
F57F  23          inc hl
F580  78          ld a,b
F581  B1          or c
```

```
F582  C4 F2 FF    call nz,FFF2     ldir
F585  2B          dec hl
F586  22 8D B0    ld (B08D),hl     low end of used string space pointer
F589  EB          ex de,hl
F58A  22 89 AE    ld (AE89),hl     upper end of DIM'd variables pointer
F58D  C9          ret

----- reset BASIC STACK
      @ C183! F5C4!
F58E  F5          push af
F58F  E5           push hl
F590  21 8B AE      ld hl,AE8B     start of BASIC STACK
F593  22 8B B0      ld (B08B),hl   BASIC STACK pointer
F596  3E 01         ld a,01
F598  CD B0 F5      call F5B0      inc BASIC STACK pointer by <a>, (hl)=next lo
F59B  36 00         ld (hl),00
F59D  E1           pop hl
F59E  F1          pop af
F59F  C9          ret

----- decrement BASIC STACK pointer by <a>
      @ C7B3! CF51! D200! D23D! D831! D8BE! F8E7  F901!
F5A0  2A 8B B0    ld hl,(B08B)     BASIC STACK pointer
F5A3  2F          cpl
F5A4  3C          inc a
F5A5  C8          ret z
F5A6  85          add a,l
F5A7  6F          ld l,a
F5A8  3E FF       ld a,FF
F5AA  8C          adc a,h
F5AB  67          ld h,a

----- set BASIC STACK pointer to <hl>
      @ C53B! C607! C628! C713! C753! C785! CAAA! D92F! D976! DA34!
F5AC  22 8B B0    ld (B08B),hl     BASIC STACK pointer
F5AF  C9          ret

----- inc BASIC STACK pointer by <a>, (hl)=next loc, check ovfl
      @ C574! C6F9! C758! D86A! D961! DA12! DA4E! DA6A! F598! FF59!
F5B0  2A 8B B0    ld hl,(B08B)     BASIC STACK pointer
F5B3  E5          push hl
F5B4  85           add a,l
F5B5  6F           ld l,a
F5B6  8C           adc a,h         add <a> to <hl>
F5B7  95           sub l
F5B8  67           ld h,a
F5B9  22 8B B0     ld (B08B),hl    BASIC STACK pointer
F5BC  3E 78        ld a,78         <hl>+4F78
F5BE  85           add a,l
F5BF  3E 4F        ld a,4F         check whether there is stll space for
F5C1  8C           adc a,h           one more [FOR]-entry on the stack
F5C2  E1          pop hl
F5C3  D0          ret nc
F5C4  CD 8E F5    call F58E        reset BASIC STACK
F5C7  C3 3E F7    jp F73E          Error: Memory full

----- set low string end up to himem
      @ C18E!
F5CA  2A 8F B0    ld hl,(B08F)     upper bound for string space pointer
F5CD  22 8D B0    ld (B08D),hl     low end of used string space pointer
F5D0  C9          ret
```

```
----- allocate space for new string =(hl)low end
      @ FC1D!
F5D1  2F          cpl
F5D2  4F          ld c,a
F5D3  06 FF       ld b,FF          <bc> is the complement of the size to alloca
F5D5  03          inc bc
F5D6  CD E6 F5    call F5E6        try allocate space for new string
F5D9  D0          ret nc
F5DA  CD 3E FC    call FC3E        GARBAGE COLLECT
F5DD  CD E6 F5    call F5E6        try allocate space for new string
F5E0  D0          ret nc
F5E1  1E 0E       ld e,0E          String space full
F5E3  C3 94 CA    jp CA94          perform ERROR <e> routine

----- try allocate space for new string
      @ F5D6! F5DD!
F5E6  2A 89 AE    ld hl,(AE89)     upper end of DIM'd variables pointer
F5E9  EB          ex de,hl
F5EA  2A 8D B0    ld hl,(B08D)     low end of used string space pointer
F5ED  09          add hl,bc
F5EE  CD B8 FF    call FFB8        test HL=DE? (try hl-de)
F5F1  D8          ret c
F5F2  22 8D B0    ld (B08D),hl     low end of used string space pointer
F5F5  23          inc hl
F5F6  EB          ex de,hl
F5F7  C9          ret

----- allocate space for new variables
      @ D755! D89B! E6F0!
F5F8  2A 89 AE    ld hl,(AE89)     upper end of DIM'd variables pointer

      @ D8D9!
F5FB  C5          push bc
F5FC  D5           push de
F5FD  D5            push de
F5FE  E5             push hl
F5FF  CD 18 F6        call F618    add hl=hl+bc, check ovfl
F602  DA 3E F7        jp c,F73E    Error: Memory full
F605  E1             pop hl
F606  C1            pop bc
F607  D5            push de
F608  7D             ld a,l
F609  91             sub c
F60A  4F             ld c,a
F60B  7C             ld a,h
F60C  98             sbc a,b
F60D  47             ld b,a
F60E  2B             dec hl
F60F  1B             dec de
F610  B1             or c
F611  C4 F5 FF       call nz,FFF5  lddr
F614  E1            pop hl
F615  D1           pop de
F616  C1          pop bc
F617  C9          ret

----- add hl=hl+bc, check ovfl
      @ F5FF!
F618  09          add hl,bc
F619  D8          ret c
F61A  EB          ex de,hl
F61B  CD 22 F6    call F622        get low end of string space, hl=de?
F61E  D0          ret nc
F61F  CD 3E FC    call FC3E        GARBAGE COLLECT
```

```
----- get low end of string space, hl=de?
      @ F61B!
F622  2A 8D B0    ld hl,(B08D)      low end of used string space pointer
F625  C3 B8 FF    jp FFB8           test HL=DE? (try hl-de)

----- space left between low string/upper DIM'd VAR
      @ FC38!
F628  2A 89 AE    ld hl,(AE89)      upper end of DIM'd variables pointer
F62B  EB          ex de,hl
F62C  2A 8D B0    ld hl,(B08D)      low end of used string space pointer
F62F  C3 CF FF    jp FFCF           HL=HL-DE

----- allocate tape buffer for input
      @ D26D!
F632  11 01 00    ld de,0001        <e>=mark for input
F635  18 03       jr F63A

----- allocate a tape buffer for output
      @ D24B! D259!
F637  11 02 08    ld de,0802        <e>=mark for output
F63A  C5          push bc
F63B  E5           push hl
F63C  21 91 B0      ld hl,B091      tape buffer flag
F63F  7E            ld a,(hl)
F640  B7            or a
F641  20 1D         jr nz,F660      ...
F643  D5            push de
F644  E5             push hl
F645  21 00 10        ld hl,1000    size of buffer 2k
F648  01 00 00        ld bc,0000
F64B  CD 43 F7        call F743     decrease HIMEM by <de>; below <bc>=error
F64E  22 92 B0        ld (B092),hl pointer to tape buffer, lower end
F651  EB              ex de,hl
F652  2A 7D AE        ld hl,(AE7D) himem for SYMBOL AFTER pointer
F655  22 94 B0        ld (B094),hl pointer to tape buffer, upper end
F658  EB              ex de,hl
F659  22 7D AE        ld (AE7D),hl himem for SYMBOL AFTER pointer
F65C  E1             pop hl
F65D  D1            pop de
F65E  3E 04         ld a,04         =4.
F660  B3            or e
F661  77            ld (hl),a
F662  2A 92 B0      ld hl,(B092)    pointer to tape buffer, lower end
F665  23            inc hl
F666  1E 00         ld e,00
F668  19            add hl,de
F669  EB            ex de,hl
F66A  E1           pop hl
F66B  C1          pop bc
F66C  C9          ret

----- release tape input buffer
      @ D29C! D2B3!
F66D  3E FE       ld a,FE           mark input buffer
F66F  18 06       jr F677

----- release tape output buffer
      @ D251! D2A8! D2B9!
F671  3E FD       ld a,FD           mark output buffer
F673  18 02       jr F677
```

```
----- release buffer
      @ F4F9! F6F7!
F675  3E FF       ld a,FF           mark any buffer
F677  C5          push bc
F678  D5           push de
F679  E5            push hl
F67A  21 91 B0       ld hl,B091     tape buffer flag
F67D  A6             and (hl)       test buffer type
F67E  77             ld (hl),a
F67F  FE 04          cp 04          =4.
F681  20 16          jr nz,F699     ...
F683  2A 92 B0       ld hl,(B092)   pointer to tape buffer, lower end
F686  EB             ex de,hl
F687  21 00 10       ld hl,1000     size of buffer
F68A  CD 2E F7       call F72E      ...
F68D  20 0A          jr nz,F699     ...
F68F  AF             xor a          reset
F690  32 91 B0       ld (B091),a    tape buffer flag
F693  2A 94 B0       ld hl,(B094)   pointer to tape buffer, upper end
F696  22 7D AE       ld (AE7D),hl   himem for SYMBOL AFTER pointer
F699  E1            pop hl
F69A  D1           pop de
F69B  C1          pop bc
F69C  C9          ret

----- command: SYMBOL <symbol#>,<list of<parameter>>
      @ DE9F
F69D  FE 80       cp 80             [AFTER]
F69F  28 2C       jr z,F6CD         command: SYMBOL AFTER <first symbol#>
F6A1  CD 67 CE    call CE67         get byte VAL(expression) in <de>
F6A4  4F          ld c,a
F6A5  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
F6A8  2C          ',
F6A9  06 08       ld b,08           count 8 arguments
F6AB  CD 67 CE    call CE67         get byte VAL(expression) in <de>
F6AE  F5          push af           save arguments on stack
F6AF  05           dec b
F6B0  28 08        jr z,F6BA        all done
F6B2  CD 55 DD     call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
F6B5  38 F4        jr c,F6AB        next argument
F6B7  AF           xor a
F6B8  18 F4        jr F6AE          replace missing arguments by 0

F6BA  EB           ex de,hl
F6BB  79           ld a,c
F6BC  CD A5 BB     call BBA5        TXT GET char <a> MATRIX, (hl)=address, carry
F6BF  30 68        jr nc,F729       Error: Improper argument
F6C1  01 08 00     ld bc,0008       count
F6C4  09           add hl,bc        add 8 for later decrement
F6C5  F1          pop af            get argument from stack
F6C6  2B          dec hl
F6C7  77          ld (hl),a         store as user matrix byte
F6C8  0D          dec c
F6C9  20 FA       jr nz,F6C5        next argument
F6CB  EB          ex de,hl
F6CC  C9          ret

----- command: SYMBOL AFTER <first symbol#>
F6CD  CD 3F DD    call DD3F         CHRGET <a>, skip blank, cp 01
F6D0  CD 86 CE    call CE86         get integer VAL(expression) in <de>
F6D3  E5          push hl
F6D4  21 00 01     ld hl,0100       maximum VAL
F6D7  CD B8 FF     call FFB8        test HL=DE? (try hl-de)
F6DA  38 4D        jr c,F729        Error: Improper argument
F6DC  D5           push de
```

```
F6DD  CD AE BB      call BBAE       TXT GET user MATRIX TABLE (hl)=addr, <a>=fir
F6E0  EB            ex de,hl
F6E1  30 1D         jr nc,F700      ...
F6E3  2F            cpl
F6E4  6F            ld l,a
F6E5  26 00         ld h,00
F6E7  23            inc hl
F6E8  29            add hl,hl
F6E9  29            add hl,hl
F6EA  29            add hl,hl       * 8 bytes per symbol
F6EB  1B            dec de
F6EC  CD 2E F7      call F72E       ...
F6EF  20 38         jr nz,F729      Error: Improper argument
F6F1  2A 96 B0      ld hl,(B096)    himem for SYMBOL AFTER (SYS)
F6F4  22 7D AE      ld (AE7D),hl    himem for SYMBOL AFTER pointer
F6F7  CD 75 F6      call F675       release buffer
F6FA  11 00 01      ld de,0100      ...
F6FD  CD AB BB      call BBAB       TXT SET user MATRIX TABLE addr (de), (hl)=ne
F700  D1            pop de
F701  CD 06 F7      call F706       set SYMBOL AFTER <de>
F704  E1            pop hl
F705  C9            ret

----- set SYMBOL AFTER <de>
      @ C03A! F701!
F706  AF            xor a           =0
F707  93            sub e
F708  6F            ld l,a
F709  3E 01         ld a,01         <l>=0-<e>
F70B  9A            sbc a,d
F70C  67            ld h,a          <h>=1-<d>-carry
F70D  B5            or l
F70E  C8            ret z           nothing to do
F70F  D5            push de
F710  29            add hl,hl       *2
F711  29            add hl,hl       *4
F712  29            add hl,hl       *8 bytes per symbol
F713  01 00 40      ld bc,4000      symbol image source, upper end
F716  CD 43 F7      call F743       decrease HIMEM by <de>; below <bc>=error
F719  EB            ex de,hl
F71A  2A 7D AE      ld hl,(AE7D)    himem for SYMBOL AFTER pointer
F71D  22 96 B0      ld (B096),hl    himem for SYMBOL AFTER (SYS)
F720  EB            ex de,hl
F721  22 7D AE      ld (AE7D),hl    himem for SYMBOL AFTER pointer
F724  D1            pop de
F725  23            inc hl
F726  C3 AB BB      jp BBAB         TXT SET user MATRIX TABLE addr (de), (hl)=ne

----- Error: Improper argument
F729  1E 05         ld e,05         Improper argument
F72B  C3 94 CA      jp CA94         perform ERROR <e> routine

      @ F68A! F6EC!
F72E  E5            push hl
F72F  2A 7B AE      ld hl,(AE7B)    himem for Basic pointer
F732  CD B8 FF      call FFB8       test HL=DE? (try hl-de)
F735  E1            pop hl
F736  C0            ret nz
F737  19            add hl,de
F738  22 7D AE      ld (AE7D),hl    himem for SYMBOL AFTER pointer
F73B  EB            ex de,hl
F73C  18 12         jr F750
```

```
----- Error: Memory full
      @ F51A  F5C7  F602  F74D' F75A' F763' F769'
F73E  1E 07          ld e,07           Memory full
F740  C3 94 CA       jp CA94           perform ERROR <e> routine

----- decrease HIMEM by <de>; below <bc>=error
      @ F64B! F716!
F743  EB             ex de,hl
F744  2A 7B AE       ld hl,(AE7B)      himem for Basic pointer
F747  CD CF FF       call FFCF         HL=HL-DE
F74A  CD BE FF       call FFBE         test HL=BC? (try hl-bc)
F74D  38 EF          jr c,F73E         Error: Memory full
F74F  EB             ex de,hl

      @ F4F6! F73C'
F750  CD 3E FC       call FC3E         GARBAGE COLLECT
F753  D5             push de
F754  2A 7D AE        ld hl,(AE7D)     himem for SYMBOL AFTER pointer
F757  CD B8 FF        call FFB8        test HL=DE? (try hl-de)
F75A  38 E2           jr c,F73E        Error: Memory full
F75C  CD 1D F5        call F51D        <bc>= space occupied by strings
F75F  2A 89 AE        ld hl,(AE89)     upper end of DIM'd variables pointer
F762  09              add hl,bc
F763  38 D9           jr c,F73E        Error: Memory full
F765  2B              dec hl
F766  CD B8 FF        call FFB8        test HL=DE? (try hl-de)
F769  30 D3           jr nc,F73E       Error: Memory full
F76B  2A 7B AE        ld hl,(AE7B)     himem for Basic pointer
F76E  EB              ex de,hl
F76F  CD CF FF        call FFCF        HL=HL-DE
F772  22 98 B0        ld (B098),hl     used by GARBAGE COLLECT
F775  11 BB F7        ld de,F7BB       ...
F778  CD 74 DA        call DA74        ...
F77B  ED 4B 98 B0     ld bc,(B098)     used by GARBAGE COLLECT
F77F  78              ld a,b
F780  07              rlca
F781  38 16           jr c,F799        ...
F783  B1              or c
F784  28 2F           jr z,F7B5        ...
F786  2A 8F B0        ld hl,(B08F)     upper bound for string space pointer
F789  54              ld d,h
F78A  5D              ld e,l
F78B  09              add hl,bc
F78C  E5              push hl
F78D  CD 1D F5         call F51D       <bc>= space occupied by strings
F790  EB               ex de,hl
F791  78               ld a,b
F792  B1               or c
F793  C4 F5 FF         call nz,FFF5    lddr
F796  E1              pop hl
F797  18 15           jr F7AE          ...

F799  2A 8D B0        ld hl,(B08D)     low end of used string space pointer
F79C  54              ld d,h
F79D  5D              ld e,l
F79E  09              add hl,bc
F79F  E5              push hl
F7A0  CD 1D F5         call F51D       <bc>= space occupied by strings
F7A3  EB               ex de,hl
F7A4  23               inc hl
F7A5  13               inc de
F7A6  78               ld a,b
F7A7  B1               or c
F7A8  C4 F2 FF         call nz,FFF2    ldir
F7AB  EB               ex de,hl
```

```
F7AC  2B          dec hl
F7AD  D1          pop de
F7AE  22 8F B0    ld (B08F),hl    upper bound for string space pointer
F7B1  EB          ex de,hl
F7B2  22 8D B0    ld (B08D),hl    low end of used string space pointer
F7B5  E1          pop hl
F7B6  22 7B AE    ld (AE7B),hl    himem for Basic pointer
F7B9  AF          xor a
F7BA  C9          ret

      @ F775:
F7BB  2A 83 AE    ld hl,(AE83)    end of BASIC program pointer
F7BE  CD BE FF    call FFBE       test HL=BC? (try hl-bc)
F7C1  D0          ret nc
F7C2  2A 98 B0    ld hl,(B098)    used by GARBAGE COLLECT
F7C5  09          add hl,bc
F7C6  EB          ex de,hl
F7C7  72          ld (hl),d
F7C8  2B          dec hl
F7C9  73          ld (hl),e
F7CA  C9          ret

----- [+]"text"; calculate len; copy temp to stack
      @ CFDA  DB95! DC31
F7CB  23          inc hl
F7CC  CD F9 F7    call F7F9       caller first! Copy temp to stack
F7CF  7E          ld a,(hl)
F7D0  FE 22       cp 22           '"
F7D2  CA 3F DD    jp z,DD3F       CHRGET <a>, skip blank, cp 01
F7D5  B7          or a            is it a zero byte?
F7D6  28 37       jr z,F80F       test previous <b> char's for SP,HT,CR,LF
F7D8  04          inc b
F7D9  23          inc hl
F7DA  18 F3       jr F7CF         get next char

----- test len of string; copy temp to stack
      @ DB0E! DC29  F240! F491!
F7DC  CD F9 F7    call F7F9       caller first! Copy temp to stack
F7DF  7E          ld a,(hl)
F7E0  B7          or a            is it a zero byte?
F7E1  C8          ret z
F7E2  23          inc hl
F7E3  04          inc b           inc len count
F7E4  18 F9       jr F7DF         next char

----- eliminate superfluous char's at string end
      @ DC35
F7E6  CD F9 F7    call F7F9       caller first! Copy temp to stack
F7E9  4F          ld c,a
F7EA  7E          ld a,(hl)
F7EB  B7          or a
F7EC  28 21       jr z,F80F       test previous <b> char's for SP,HT,CR,LF
F7EE  B9          cp c
F7EF  28 1E       jr z,F80F       test previous <b> char's for SP,HT,CR,LF
F7F1  FE 2C       cp 2C           ',
F7F3  28 1A       jr z,F80F       test previous <b> char's for SP,HT,CR,LF
F7F5  23          inc hl
F7F6  04          inc b
F7F7  18 F1       jr F7EA         test next char

----- caller first! Copy temp to stack
      @ F7CC! F7DC! F7E6!
F7F9  D1          pop de          get return address
F7FA  E5          push hl
F7FB  06 00       ld b,00         initial string len count
```

```
F7FD  CD FB FF     call FFFB       jp(de); do calling routine first, return her
F800  D1          pop de
F801  E5          push hl
F802  21 BA B0     ld hl,B0BA      temporary string descriptor
F805  70           ld (hl),b       =len
F806  23           inc hl
F807  73           ld (hl),e
F808  23           inc hl          =address
F809  72           ld (hl),d
F80A  CD BA FB     call FBBA       copy string descr to string stack, check ovf
F80D  E1          pop hl
F80E  C9          ret

----- test previous <b> char's for SP,HT,CR,LF
      @ F7D6' F7EC' F7EF' F7F3'
F80F  E5          push hl
F810  04           inc b
F811  05           dec b
F812  28 12        jr z,F826       all done
F814  2B           dec hl
F815  7E           ld a,(hl)
F816  FE 20        cp 20           'SPACE
F818  28 F7        jr z,F811
F81A  FE 09        cp 09           'HT  (^I)
F81C  28 F3        jr z,F811
F81E  FE 0D        cp 0D           'CR  (^M)
F820  28 EF        jr z,F811
F822  FE 0A        cp 0A           'LF  (^J)
F824  28 EB        jr z,F811
F826  E1          pop hl
F827  C9          ret

----- print this line
      @ DB9C! F253! F392! F494! F49E!
F828  CD DA FB    call FBDA        try to release string (FAC); <a>=len, z=zero
F82B  C8          ret z
F82C  1A          ld a,(de)
F82D  13          inc de
F82E  CD 6E C3    call C36E        output char <a> to channel
F831  10 F9       djnz F82C        next char
F833  C9          ret

----- function: LOWER$(<string expression>)
      @ D1D0
F834  01 39 F8    ld bc,F839       change <a> to lower case
F837  18 0C       jr F845

----- change <a> to lower case
      @ F834:
F839  FE 41       cp 41            'A
F83B  D8          ret c
F83C  FE 5B       cp 5B            '[
F83E  D0          ret nc
F83F  C6 20       add a,20         'SPACE
F841  C9          ret

----- function: UPPER$(<string expression>)
      @ D1E6
F842  01 8A FF    ld bc,FF8A       change <a> to upper case
F845  C5          push bc
F846  2A C2 B0     ld hl,(B0C2)    Floating point ACU, FAC
F849  7E           ld a,(hl)
F84A  CD 19 FC     call FC19       allocate new string (ed), set VAR pointer to
F84D  D5           push de
F84E  CD DA FB      call FBDA      try to release string (FAC); <a>=len, z=zero
```

```
F851  E1          pop hl
F852  C1          pop bc
F853  3C          inc a
F854  3D          dec a
F855  CA BA FB    jp z,FBBA         copy string descr to string stack, check ovf
F858  F5          push af
F859  1A          ld a,(de)
F85A  13          inc de
F85B  CD F9 FF    call FFF9         jp(bc)
F85E  77          ld (hl),a
F85F  23          inc hl
F860  F1          pop af
F861  18 F1       jr F854           next

----- append string (hl) to string (FAC)
      @ CF2B!
F863  E5          push hl
F864  7E          ld a,(hl)
F865  2A C2 B0    ld hl,(B0C2)      Floating point ACU, FAC
F868  86          add a,(hl)
F869  1E 0F       ld e,0F           String too long
F86B  DA 94 CA    jp c,CA94         perform ERROR <e> routine
F86E  CD 19 FC    call FC19         allocate new string (ed), set VAR pointer to
F871  E1          pop hl
F872  D5          push de
F873  E5          push hl
F874  CD DA FB    call FBDA         try to release string (FAC); <a>=len, z=zero
F877  48          ld c,b
F878  EB          ex de,hl
F879  E3          ex (sp),hl
F87A  CD E8 FB    call FBE8         try to release string at low end of string s
F87D  E1          pop hl
F87E  E3          ex (sp),hl
F87F  78          ld a,b
F880  CD 8B F8    call F88B         copy <a> bytes from (de) to (hl)
F883  D1          pop de
F884  79          ld a,c
F885  CD 8B F8    call F88B         copy <a> bytes from (de) to (hl)
F888  C3 BA FB    jp FBBA           copy string descr to string stack, check ovf

----- copy <a> bytes from (de) to (hl)
      @ F880! F885!
F88B  C5          push bc
F88C  EB          ex de,hl
F88D  4F          ld c,a
F88E  06 00       ld b,00           reset upper byte
F890  B7          or a
F891  C4 F2 FF    call nz,FFF2      ldir
F894  EB          ex de,hl
F895  C1          pop bc
F896  C9          ret

----- compare string (hl) with string (de)
      @ CF78!
F897  E5          push hl
F898  CD DA FB    call FBDA         try to release string (FAC); <a>=len, z=zero
F89B  48          ld c,b
F89C  E1          pop hl
F89D  D5          push de
F89E  CD E8 FB    call FBE8         try to release string at low end of string s
F8A1  E1          pop hl
F8A2  78          ld a,b
F8A3  B1          or c
F8A4  C8          ret z             both strings zero len; equal
F8A5  79          ld a,c
```

```
F8A6  B7          or a
F8A7  28 0C       jr z,F8B5        string (hl) zero len; not equal
F8A9  78          ld a,b
F8AA  B7          or a
F8AB  28 09       jr z,F8B6        string (de) zero len; it's the smaller one
F8AD  05          dec b
F8AE  0D          dec c
F8AF  1A          ld a,(de)
F8B0  13          inc de
F8B1  BE          cp (hl)          compare two string bytes
F8B2  23          inc hl
F8B3  28 ED       jr z,F8A2        equal, check len now and continue
F8B5  3F          ccf
F8B6  9F          sbc a,a
F8B7  C0          ret nz
F8B8  3C          inc a
F8B9  C9          ret

----- function: BIN$(<unsigned integer>[,<digits>])
      @ D190
F8BA  CD CE F8    call F8CE        get (EXPRESSION) and next arg in b
F8BD  D5          push de
F8BE  CD 14 F1     call F114       edit value in binary representation
F8C1  EB           ex de,hl
F8C2  18 5E        jr F922         allocate new string; copy (hl) to new (de)

----- function: HEX$(<unsigned integer>[,<digits>])
      @ D194
F8C4  CD CE F8    call F8CE        get (EXPRESSION) and next arg in b
F8C7  D5          push de
F8C8  CD 19 F1     call F119       edit value in HEX representation
F8CB  EB           ex de,hl
F8CC  18 54        jr F922         allocate new string; copy (hl) to new (de)

----- get (EXPRESSION) and next arg in b
      @ F8BA! F8C4!
F8CE  CD FB CE    call CEFB        evaluate (expression), CHRGET, cp 01
F8D1  CD 53 FF    call FF53        get VARTYPE, copy FAC to BASIC STACK
F8D4  CD 55 DD    call DD55        CHRBACK comma?; if=:CHRGET <a>, scf
F8D7  9F          sbc a,a
F8D8  DC 67 CE    call c,CE67      get byte VAL(expression) in <de>
F8DB  FE 11       cp 11            max # of digits =16.
F8DD  D2 9C FA    jp nc,FA9C       Error: improper argument
F8E0  47          ld b,a           b = # of digits to edit
F8E1  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F8E4  29          ')
F8E5  EB          ex de,hl
F8E6  79          ld a,c
F8E7  C3 A0 F5    jp F5A0          decrement BASIC STACK pointer by <a>

----- function: DEC$(<num VAR>,<string VAR>)
      @ D192
F8EA  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F8ED  28          '(
F8EE  CD FB CE    call CEFB        evaluate (expression), CHRGET, cp 01
F8F1  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F8F4  2C          ',
F8F5  CD 53 FF    call FF53        get VARTYPE, copy FAC to BASIC STACK
F8F8  CD 9F CE    call CE9F        evaluate expression, release string again
F8FB  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F8FE  29          ')
F8FF  E5          push hl
F900  79           ld a,c
F901  CD A0 F5     call F5A0       decrement BASIC STACK pointer by <a>
F904  D5           push de
```

```
F905  79              ld a,c
F906  CD 4B FF        call FF4B       set VARTYPE <a>, copy VARIABLE (hl) to FAC
F909  D1              pop de
F90A  78              ld a,b
F90B  B7              or a
F90C  C4 BA F3        call nz,F3BA    ...
F90F  30 0A           jr nc,F91B      Error: Improper argument
F911  78              ld a,b
F912  B7              or a
F913  20 06           jr nz,F91B      Error: Improper argument
F915  79              ld a,c
F916  CD 9F EE        call EE9F       ...
F919  18 07           jr F922         allocate new string; copy (hl) to new (de)

----- Error: Improper argument
F91B  C3 9C FA        jp FA9C         Error: improper argument

----- function: STR$(<numeric expression>)
      @ D1E0
F91E  E5              push hl
F91F  CD 9D EE        call EE9D       ...

----- allocate new string; copy (hl) to new (de)
      @ F8C2' F8CC' F919'
F922  E5              push hl
F923  01 FF FF        ld bc,FFFF      initial len count = -1
F926  03              inc bc
F927  7E              ld a,(hl)
F928  23              inc hl
F929  B7              or a            check len of string to copy
F92A  20 FA           jr nz,F926      string not terminated, check next byte
F92C  79              ld a,c
F92D  CD 19 FC        call FC19       allocate new string (ed), set VAR pointer to
F930  E1              pop hl
F931  B7              or a
F932  D5              push de
F933  C4 F2 FF        call nz,FFF2    ldir
F936  D1              pop de
F937  CD BA FB        call FBBA       copy string descr to string stack, check ovf
F93A  E1              pop hl
F93B  C9              ret

----- function: LEFT$(<string expression>,<len>)
      @ D198
F93C  CD E9 F9        call F9E9       get string expression (de) and byte <a>
F93F  0E 00           ld c,00
F941  18 2A           jr F96D         ...

----- function: RIGHT$(<string expression>,<len>)
      @ D1A0
F943  CD E9 F9        call F9E9       get string expression (de) and byte <a>
F946  1A              ld a,(de)
F947  90              sub b
F948  4F              ld c,a
F949  18 22           jr F96D         ...

----- function: MID$(<string expression>,<position>[,<len>])
      @ D007
F94B  CD 37 DD        call DD37       CHRNEXT <a>, nz=Error; CHRGET
F94E  28              '(
F94F  CD E9 F9        call F9E9       get string expression (de) and byte <a>
F952  78              ld a,b
F953  B7              or a
F954  CA 9C FA        jp z,FA9C       Error: improper argument
F957  05              dec b
```

```
F958  48         ld c,b
F959  D5         push de
F95A  C5          push bc
F95B  CD FB F9     call F9FB        ...
F95E  C1          pop bc
F95F  E3          ex (sp),hl
F960  7E          ld a,(hl)
F961  91          sub c
F962  06 00       ld b,00          'NUL
F964  38 05       jr c,F96B        ...
F966  BB          cp e
F967  47          ld b,a
F968  38 01       jr c,F96B        ...
F96A  43          ld b,e
F96B  EB          ex de,hl
F96C  E1         pop hl
F96D  CD 37 DD   call DD37         CHRNEXT <a>, nz=Error; CHRGET
F970  29          ')

      @ F38F!
F971  E5         push hl
F972  EB          ex de,hl
F973  7E          ld a,(hl)
F974  B8          cp b
F975  78          ld a,b
F976  30 03       jr nc,F97B       ...
F978  7E          ld a,(hl)
F979  0E 00       ld c,00
F97B  F5          push af
F97C  CD 19 FC     call FC19       allocate new string (ed), set VAR pointer to
F97F  D5           push de
F980  CD E8 FB      call FBE8      try to release string at low end of string s
F983  EB            ex de,hl
F984  D1           pop de
F985  06 00        ld b,00
F987  09           add hl,bc
F988  F1          pop af
F989  4F          ld c,a
F98A  B7          or a
F98B  C4 F2 FF    call nz,FFF2     ldir
F98E  CD BA FB    call FBBA        copy string descr to string stack, check ovf
F991  E1         pop hl
F992  C9         ret

----- command: MID$(<stringvar>,<startpos>,<len>)=<string expression>
      @ DE59
F993  CD 37 DD   call DD37         CHRNEXT <a>, nz=Error; CHRGET
F996  28          '(
F997  CD 86 D6   call D686         get address of VARIABLE or subscript
F99A  CD 3C FF   call FF3C         test VARTYPE for string, else error
F99D  E5         push hl
F99E  EB          ex de,hl
F99F  CD 21 FB    call FB21        ...
F9A2  E3          ex (sp),hl
F9A3  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F9A6  2C          ',
F9A7  CD 6D CE    call CE6D        <a>=next VAL, 0=error
F9AA  47          ld b,a
F9AB  CD FB F9    call F9FB        ...
F9AE  4B          ld c,e
F9AF  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F9B2  29          ')
F9B3  CD 37 DD    call DD37        CHRNEXT <a>, nz=Error; CHRGET
F9B6  EF          [=]
F9B7  C5          push bc
```

```
F9B8  CD 9F CE     call CE9F        evaluate expression, release string again
F9BB  78           ld a,b
F9BC  C1           pop bc
F9BD  E3           ex (sp),hl
F9BE  0C           inc c
F9BF  0D           dec c
F9C0  28 25        jr z,F9E7        ...
F9C2  F5           push af
F9C3  7E           ld a,(hl)
F9C4  90           sub b
F9C5  DA 9C FA     jp c,FA9C        Error: improper argument
F9C8  3C           inc a
F9C9  B9           cp c
F9CA  38 01        jr c,F9CD        ...
F9CC  79           ld a,c
F9CD  4F           ld c,a
F9CE  78           ld a,b
F9CF  3D           dec a
F9D0  23           inc hl
F9D1  86           add a,(hl)
F9D2  23           inc hl
F9D3  66           ld h,(hl)
F9D4  6F           ld l,a
F9D5  8C           adc a,h
F9D6  95           sub l
F9D7  67           ld h,a
F9D8  F1           pop af
F9D9  47           ld b,a
F9DA  EB           ex de,hl
F9DB  79           ld a,c
F9DC  B8           cp b
F9DD  38 01        jr c,F9E0        ...
F9DF  78           ld a,b
F9E0  4F           ld c,a
F9E1  06 00        ld b,00
F9E3  B7           or a
F9E4  C4 F2 FF     call nz,FFF2     ldir
F9E7  E1           pop hl
F9E8  C9           ret

----- get string expression (de) and byte <a>
      @ F93C! F943! F94F!
F9E9  CD A5 CE     call CEA5        evaluate (string expression)
F9EC  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
F9EF  2C           ',
F9F0  E5           push hl
F9F1  2A C2 B0     ld hl,(B0C2)     Floating point ACU, FAC
F9F4  E3           ex (sp),hl
F9F5  CD 67 CE     call CE67        get byte VAL(expression) in <de>
F9F8  47           ld b,a
F9F9  D1           pop de
F9FA  C9           ret

      @ F95B! F9AB!
F9FB  1E FF        ld e,FF          =255.
F9FD  7E           ld a,(hl)
F9FE  FE 29        cp 29            ')
FA00  C8           ret z
FA01  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
FA04  2C           ',
FA05  CD 67 CE     call CE67        get byte VAL(expression) in <de>
FA08  5F           ld e,a
FA09  C9           ret
```

```
----- function: LEN(<string expression>)
      @ D1CA
FA0A  CD DA FB    call FBDA         try to release string (FAC); <a>=len, z=zero
FA0D  C3 0A FF    jp FF0A           set FAC to <a> and mark integer

----- function: ASC(<string expression>)
      @ D1B0
FA10  CD 70 FA    call FA70         ...
FA13  C3 0A FF    jp FF0A           set FAC to <a> and mark integer

----- function: CHR$(<byte value>)
      @ D1B4
FA16  CD 92 FA    call FA92         convert FAC to 1 byte in <A>

      @ FA2D'
FA19  F5          push af
FA1A  3E 01        ld a,01          'SOH (^A)
FA1C  CD 19 FC     call FC19        allocate new string (ed), set VAR pointer to
FA1F  F1          pop af
FA20  12          ld (de),a
FA21  C3 BA FB    jp FBBA           copy string descr to string stack, check ovf

----- function: INKEY$
      @ D0D0:
FA24  E5          push hl
FA25  CD 2A FA     call FA2A        ...
FA28  E1          pop hl
FA29  C9          ret

      @ FA25!
FA2A  CD 39 C4    call C439         jp KM read char from keyboard
FA2D  38 EA       jr c,FA19         ...
FA2F  AF          xor a
FA30  32 BA B0    ld (B0BA),a       temporary string descriptor
FA33  C3 BA FB    jp FBBA           copy string descr to string stack, check ovf

----- function: STRING$(<repeat>,<character>)
      @ D1A4
FA36  CD 67 CE    call CE67         get byte VAL(expression) in <de>
FA39  4F          ld c,a
FA3A  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
FA3D  2C          ',
FA3E  CD FB CE    call CEFB         evaluate (expression), CHRGET, cp 01
FA41  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
FA44  29          ')
FA45  E5          push hl
FA46  CD 45 FF     call FF45        get VARTYPE <a>; cp string
FA49  28 05        jr z,FA50
FA4B  CD 92 FA     call FA92        convert FAC to 1 byte in <A>
FA4E  18 03        jr FA53

FA50  CD 70 FA     call FA70        ...
FA53  41           ld b,c
FA54  4F           ld c,a
FA55  18 07        jr FA5E          ...

----- function: SPACE$(<# of spaces>)
      @ D1DA
FA57  CD 92 FA    call FA92         convert FAC to 1 byte in <A>
FA5A  47          ld b,a
FA5B  0E 20       ld c,20           'SPACE
FA5D  E5          push hl
FA5E  78           ld a,b
FA5F  CD 19 FC     call FC19        allocate new string (ed), set VAR pointer to
FA62  04           inc b
```

```
FA63  05           dec b
FA64  28 05        jr z,FA6B       ...
FA66  79           ld a,c
FA67  12           ld (de),a
FA68  13           inc de
FA69  18 F8        jr FA63         ...

FA6B  CD BA FB     call FBBA       copy string descr to string stack, check ovf
FA6E  E1          pop hl
FA6F  C9          ret

      @ FA10! FA50!
FA70  CD DA FB    call FBDA        try to release string (FAC); <a>=len, z=zero
FA73  28 27       jr z,FA9C        Error: improper argument
FA75  1A          ld a,(de)
FA76  C9          ret

----- function: VAL(<string expression>)
      @ D1E8
FA77  CD DA FB    call FBDA        try to release string (FAC); <a>=len, z=zero
FA7A  CA 0A FF    jp z,FF0A        set FAC to <a> and mark integer
FA7D  EB          ex de,hl
FA7E  E5          push hl
FA7F  5F           ld e,a
FA80  16 00        ld d,00
FA82  19           add hl,de
FA83  5E           ld e,(hl)
FA84  72           ld (hl),d
FA85  E3           ex (sp),hl
FA86  D5           push de
FA87  CD A3 EC      call ECA3      get either HEX or integer VAL
FA8A  D1           pop de
FA8B  E1          pop hl
FA8C  73          ld (hl),e
FA8D  D8          ret c
FA8E  1E 0D       ld e,0D          Type mismatch
FA90  18 0C       jr FA9E          error message, READY

----- convert FAC to 1 byte in <A>
      @ FA16! FA4B! FA57! FAAB!
FA92  E5          push hl
FA93  CD 8D FE     call FE8D       function: CINT(<num expression>) in <hl>
FA96  EB           ex de,hl
FA97  E1          pop hl
FA98  7A          ld a,d
FA99  B7          or a
FA9A  7B          ld a,e
FA9B  C8          ret z

----- Error: improper argument
FA9C  1E 05       ld e,05          Improper argument

----- error message, READY
FA9E  C3 94 CA    jp CA94          perform ERROR <e> routine

----- function: INSTR([<start >,]<string expr>,<searched string>)
      @ D196
FAA1  CD FB CE    call CEFB        evaluate (expression), CHRGET, cp 01
FAA4  CD 45 FF    call FF45        get VARTYPE <a>; cp string
FAA7  0E 01       ld c,01          default <start>
FAA9  28 0F       jr z,FABA        argument not present
FAAB  CD 92 FA    call FA92        convert FAC to 1 byte in <A>
FAAE  B7          or a             is it zero? that's an error
FAAF  CA 9C FA    jp z,FA9C        Error: improper argument
FAB2  4F          ld c,a
```

```
FAB3  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
FAB6  2C          ',
FAB7  CD A5 CE    call CEA5         evaluate (string expression)
FABA  CD 37 DD    call DD37         CHRNEXT <a>, nz=Error; CHRGET
FABD  2C          ',
FABE  E5          push hl
FABF  2A C2 B0     ld hl,(B0C2)     Floating point ACU, FAC
FAC2  E3           ex (sp),hl
FAC3  CD 9F CE     call CE9F        evaluate expression, release string again
FAC6  CD 37 DD     call DD37        CHRNEXT <a>, nz=Error; CHRGET
FAC9  29           ')
FACA  E3           ex (sp),hl
FACB  79           ld a,c
FACC  CD D4 FA     call FAD4        ...
FACF  CD 0A FF     call FF0A        set FAC to <a> and mark integer
FAD2  E1          pop hl
FAD3  C9          ret

      @ FACC!
FAD4  F5          push af
FAD5  48           ld c,b
FAD6  D5           push de
FAD7  CD E8 FB      call FBE8       try to release string at low end of string s
FADA  E1           pop hl
FADB  F1          pop af
FADC  E5          push hl
FADD  6F           ld l,a
FADE  60           ld h,b
FADF  78           ld a,b
FAE0  BD           cp l
FAE1  38 2D        jr c,FB10        ...
FAE3  2D           dec l
FAE4  7D           ld a,l
FAE5  83           add a,e
FAE6  5F           ld e,a
FAE7  8A           adc a,d
FAE8  93           sub e
FAE9  57           ld d,a
FAEA  78           ld a,b
FAEB  95           sub l
FAEC  47           ld b,a
FAED  79           ld a,c
FAEE  D6 01        sub 01
FAF0  7D           ld a,l
FAF1  3C           inc a
FAF2  38 1D        jr c,FB11        ...
FAF4  E3           ex (sp),hl
FAF5  C5           push bc
FAF6  D5            push de
FAF7  E5             push hl
FAF8  1A              ld a,(de)
FAF9  BE              cp (hl)
FAFA  20 0D           jr nz,FB09    ...
FAFC  23              inc hl
FAFD  0D              dec c
FAFE  28 13           jr z,FB13     ...
FB00  13              inc de
FB01  05              dec b
FB02  20 F4           jr nz,FAF8    ...
FB04  E1             pop hl
FB05  D1            pop de
FB06  C1           pop bc
FB07  18 07        jr FB10          ...
```

```
FB09  E1           pop hl
FB0A  D1           pop de
FB0B  C1           pop bc
FB0C  13           inc de
FB0D  05           dec b
FB0E  20 E5        jr nz,FAF5        ...
FB10  AF           xor a
FB11  D1           pop de
FB12  C9           ret

FB13  E1           pop hl
FB14  D1           pop de
FB15  C1           pop bc
FB16  E1           pop hl
FB17  7C           ld a,h
FB18  90           sub b
FB19  3C           inc a
FB1A  C9           ret

      @ EA6E!
FB1B  11 2E FB     ld de,FB2E        ...
FB1E  C3 74 DA     jp DA74           ...

      @ F99F!
FB21  E5           push hl
FB22  7E           ld a,(hl)
FB23  23           inc hl
FB24  4E           ld c,(hl)
FB25  23           inc hl
FB26  46           ld b,(hl)
FB27  EB           ex de,hl
FB28  B7           or a
FB29  C4 2E FB     call nz,FB2E      ...
FB2C  E1           pop hl
FB2D  C9           ret

      @ FB1B: FB29!
FB2E  2A 8D B0     ld hl,(B08D)      low end of used string space pointer
FB31  CD BE FF     call FFBE         test HL=BC? (try hl-bc)
FB34  30 07        jr nc,FB3D        ...
FB36  2A 8F B0     ld hl,(B08F)      upper bound for string space pointer
FB39  CD BE FF     call FFBE         test HL=BC? (try hl-bc)
FB3C  D0           ret nc
FB3D  EB           ex de,hl
FB3E  2B           dec hl
FB3F  2B           dec hl
FB40  E5           push hl
FB41  CD 8F FB     call FB8F         allocate string and copy it from (FAC)
FB44  EB           ex de,hl
FB45  E1           pop hl
FB46  C3 A6 FB     jp FBA6           copy string descriptor (de) to (hl)

      @ D188!
FB49  2A C2 B0     ld hl,(B0C2)      Floating point ACU, FAC
FB4C  11 BA B0     ld de,B0BA        temporary string descriptor
FB4F  CD B8 FF     call FFB8         test HL=DE? (try hl-de)
FB52  D8           ret c
FB53  CD 8F FB     call FB8F         allocate string and copy it from (FAC)
FB56  C3 BA FB     jp FBBA           copy string descr to string stack, check ovf

----- release string, allocate new one, copy string from (FAC)
      @ D676!
FB59  2A C2 B0     ld hl,(B0C2)      Floating point ACU, FAC
FB5C  E5           push hl
FB5D  7E           ld a,(hl)         len of string
```

```
FB5E  B7         or a
FB5F  28 26      jr z,FB87         zero len, do nothing here
FB61  23         inc hl
FB62  5E         ld e,(hl)         (ed)=string address
FB63  23         inc hl
FB64  56         ld d,(hl)
FB65  2A 81 AE   ld hl,(AE81)      start of BASIC program -1 pointer
FB68  CD B8 FF   call FFB8         test HL=DE? (try hl-de)
FB6B  30 1E      jr nc,FB8B        string below Basic program
FB6D  2A 8F B0   ld hl,(B08F)      upper bound for string space pointer
FB70  CD B8 FF   call FFB8         test HL=DE? (try hl-de)
FB73  38 16      jr c,FB8B         string outside string space
FB75  2A 83 AE   ld hl,(AE83)      end of BASIC program pointer
FB78  CD B8 FF   call FFB8         test HL=DE? (try hl-de)
FB7B  30 0A      jr nc,FB87        string within Basic program
FB7D  E1        pop hl
FB7E  E5        push hl
FB7F  11 9C B0   ld de,B09C        string stack
FB82  CD B8 FF   call FFB8         test HL=DE? (try hl-de)
FB85  20 04      jr nz,FB8B        string not on string stack
FB87  E1        pop hl
FB88  C3 FF FB  jp FBFF            try delete string from string stack

FB8B  E1        pop hl
FB8C  CD FF FB  call FBFF          try delete string from string stack

----- allocate string and copy it from (FAC)
      @ FB41! FB53!
FB8F  7E        ld a,(hl)
FB90  CD 19 FC  call FC19          allocate new string (ed), set VAR pointer to
FB93  D5        push de
FB94  4E         ld c,(hl)
FB95  06 00      ld b,00           clear most significant byte for ldir
FB97  23         inc hl
FB98  7E         ld a,(hl)
FB99  23         inc hl
FB9A  66         ld h,(hl)         (hl)=string address
FB9B  6F         ld l,a
FB9C  78         ld a,b            =0
FB9D  B1         or c              =len
FB9E  C4 F2 FF   call nz,FFF2      ldir
FBA1  D1        pop de
FBA2  21 BA B0  ld hl,B0BA         temporary string descriptor
FBA5  C9        ret

----- copy string descriptor (de) to (hl)
      @ FB46  FBD3!
FBA6  1A        ld a,(de)
FBA7  13        inc de
FBA8  77        ld (hl),a
FBA9  23        inc hl
FBAA  1A        ld a,(de)
FBAB  13        inc de
FBAC  77        ld (hl),a
FBAD  23        inc hl
FBAE  1A        ld a,(de)
FBAF  13        inc de
FBB0  77        ld (hl),a
FBB1  23        inc hl
FBB2  C9        ret
```

```
----- reset string stack
      @ C162! CAAD!
FBB3  21 9C B0    ld hl,B09C        string stack
FBB6  22 9A B0    ld (B09A),hl      pointer to start of string stack
FBB9  C9          ret

----- copy string descr to string stack, check ovfl
      @ F80A! F855  F888  F937! F98E! FA21  FA33  FA6B! FB56
FBBA  3E 03       ld a,03           <string VAR$>
FBBC  32 C1 B0    ld (B0C1),a       VARTYPE
FBBF  2A 9A B0    ld hl,(B09A)      pointer to start of string stack
FBC2  22 C2 B0    ld (B0C2),hl      Floating point ACU, FAC
FBC5  11 BA B0    ld de,B0BA        =end of string stack
FBC8  CD B8 FF    call FFB8         test HL=DE? (try hl-de)
FBCB  1E 10       ld e,10           String expression too complex
FBCD  CA 94 CA    jp z,CA94         perform ERROR <e> routine
FBD0  11 BA B0    ld de,B0BA        temporary string descriptor
FBD3  CD A6 FB    call FBA6         copy string descriptor (de) to (hl)
FBD6  22 9A B0    ld (B09A),hl      pointer to start of string stack
FBD9  C9          ret

----- try to release string (FAC); <a>=len, z=zero len
      @ CEA2  DBA0! DBE2! F828! F84E! F874! F898! FA0A! FA70! FA77! FC32!
FBDA  E5          push hl
FBDB  CD 3C FF     call FF3C        test VARTYPE for string, else error
FBDE  2A C2 B0     ld hl,(B0C2)     Floating point ACU, FAC
FBE1  CD E8 FB     call FBE8        try to release string at low end of string s
FBE4  E1          pop hl
FBE5  78          ld a,b            len of string
FBE6  B7          or a
FBE7  C9          ret

----- try to release string at low end of string space
      @ F31F! F87A! F89E! F980! FAD7! FBE1!
FBE8  CD FF FB    call FBFF         try delete string from string stack
FBEB  C0          ret nz
FBEC  D5          push de
FBED  1B           dec de
FBEE  2A 8D B0     ld hl,(B08D)     low end of used string space pointer
FBF1  CD B8 FF     call FFB8        test HL=DE? (try hl-de)
FBF4  20 07        jr nz,FBFD       string not the last created string
FBF6  58           ld e,b
FBF7  16 00        ld d,00          <de>=len of string
FBF9  19           add hl,de        adjust pointer to lower end to upper end
FBFA  22 8D B0     ld (B08D),hl     low end of used string space pointer
FBFD  D1          pop de
FBFE  C9          ret

----- try delete string from string stack
      @ FB88  FB8C! FBE8!
FBFF  E5          push hl
FC00  46           ld b,(hl)        b=len
FC01  23           inc hl
FC02  7E           ld a,(hl)
FC03  23           inc hl
FC04  66           ld h,(hl)
FC05  6F           ld l,a           hl = string address
FC06  E3           ex (sp),hl
FC07  EB           ex de,hl
FC08  2A 9A B0     ld hl,(B09A)     pointer to start of string stack
FC0B  2B           dec hl
FC0C  2B           dec hl
FC0D  2B           dec hl
FC0E  CD B8 FF     call FFB8        test HL=DE? (try hl-de)
FC11  20 03        jr nz,FC16       string was not on string stack
```

```
FC13  22 9A B0     ld (B09A),hl     pointer to start of string stack
FC16  EB           ex de,hl
FC17  D1          pop de
FC18  C9          ret

----- allocate new string (ed), set VAR pointer to it
      @ F84A! F86E! F92D! F97C! FA1C! FA5F! FB90!
FC19  F5          push af
FC1A  C5           push bc
FC1B  E5            push hl
FC1C  F5             push af
FC1D  CD D1 F5        call F5D1     allocate space for new string =(hl)low end
FC20  F1             pop af
FC21  21 BA B0       ld hl,B0BA     temporary string descriptor
FC24  77             ld (hl),a      len
FC25  23             inc hl
FC26  73             ld (hl),e
FC27  23             inc hl
FC28  72             ld (hl),d      (ed)=address of string
FC29  E1            pop hl
FC2A  C1           pop bc
FC2B  F1          pop af
FC2C  C9          ret

----- function: FRE(0), or FRE("")
      @ D1C0
FC2D  CD 45 FF    call FF45         get VARTYPE <a>; cp string
FC30  20 06       jr nz,FC38        not a string
FC32  CD DA FB    call FBDA         try to release string (FAC); <a>=len, z=zero
FC35  CD 3E FC    call FC3E         GARBAGE COLLECT
FC38  CD 28 F6    call F628         space left between low string/upper DIM'd VA
FC3B  C3 60 FE    jp FE60           convert unsigned integer (hl) to real

----- GARBAGE COLLECT
      @ EA71! F4EF! F5DA! F61F! F750! FC35!
FC3E  C5          push bc
FC3F  D5           push de
FC40  E5            push hl
FC41  2A 8F B0       ld hl,(B08F)   upper bound for string space pointer
FC44  22 8D B0       ld (B08D),hl   low end of used string space pointer
FC47  21 00 00       ld hl,0000
FC4A  22 BD B0       ld (B0BD),hl   used on GARBAGE COLLECT
FC4D  2A 89 AE       ld hl,(AE89)   upper end of DIM'd variables pointer
FC50  22 BF B0       ld (B0BF),hl   save on GARBAGE COLLECT
FC53  CD 7B FC       call FC7B      ...
FC56  2A BD B0       ld hl,(B0BD)   used on GARBAGE COLLECT
FC59  7C             ld a,h
FC5A  B5             or l
FC5B  28 1A          jr z,FC77      ...
FC5D  56             ld d,(hl)
FC5E  2B             dec hl

      @ CF8A
FC5F  5E          ld e,(hl)
FC60  E5          push hl
FC61  2B           dec hl
FC62  4E           ld c,(hl)
FC63  06 00        ld b,00
FC65  2A 8D B0     ld hl,(B08D)     low end of used string space pointer
FC68  EB           ex de,hl
FC69  09           add hl,bc
FC6A  2B           dec hl
FC6B  CD F5 FF     call FFF5        lddr
FC6E  13           inc de
FC6F  E1          pop hl
```

```
FC70  73          ld (hl),e
FC71  23          inc hl
FC72  72          ld (hl),d
FC73  1B          dec de
FC74  EB          ex de,hl
FC75  18 CD       jr FC44          ...

FC77  E1            pop hl
FC78  D1           pop de
FC79  C1          pop bc
FC7A  C9          ret

      @ FC53!
FC7B  21 9C B0    ld hl,B09C       string stack
FC7E  ED 5B 9A B0 ld de,(B09A)     pointer to start of string stack
FC82  CD B8 FF    call FFB8        test HL=DE? (try hl-de)
FC85  28 0F       jr z,FC96        ...
FC87  7E          ld a,(hl)
FC88  23          inc hl
FC89  4E          ld c,(hl)
FC8A  23          inc hl
FC8B  46          ld b,(hl)
FC8C  E5          push hl
FC8D  EB           ex de,hl
FC8E  B7           or a
FC8F  C4 9C FC     call nz,FC9C    ...
FC92  E1          pop hl
FC93  23          inc hl
FC94  18 E8       jr FC7E          ...

FC96  11 9C FC    ld de,FC9C       ...
FC99  C3 74 DA    jp DA74          ...

      @ FC8F! FC96:
FC9C  2A 8D B0    ld hl,(B08D)     low end of used string space pointer
FC9F  CD BE FF    call FFBE        test HL=BC? (try hl-bc)
FCA2  D8          ret c
FCA3  2A BF B0    ld hl,(B0BF)     save on GARBAGE COLLECT
FCA6  CD BE FF    call FFBE        test HL=BC? (try hl-bc)
FCA9  D0          ret nc
FCAA  EB          ex de,hl
FCAB  22 BD B0    ld (B0BD),hl     used on GARBAGE COLLECT
FCAE  ED 43 BF B0 ld (B0BF),bc     save on GARBAGE COLLECT
FCB2  C9          ret
```

```
      @ EEA3!
FCB3  CD 2D FF     call FF2D        <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FCB6  D2 52 BD     jp nc,BD52       REAL ARITH ??
FCB9  CD A3 BD     call BDA3        INT ARITH, ??
FCBC  22 C2 B0     ld (B0C2),hl     Floating point ACU, FAC
FCBF  21 C3 B0     ld hl,B0C3       (=FAC+1)
FCC2  C9           ret

      @ EE84!
FCC3  CD C2 FE     call FEC2        function: UNT(<address expression>)
FCC6  21 C3 B0     ld hl,B0C3       = string address
FCC9  C3 A6 BD     jp BDA6          INT ARITH, BC=0002; E=0

----- perform [+] (plus)
      @ C67B! CF82
FCCC  CD 15 FE     call FE15        compare VARTYPEs; change if <>; string illeg
FCCF  30 09        jr nc,FCDA       it's at least one real argument
FCD1  CD AC BD     call BDAC        INT ARITH ADD; <hl>=<hl>+<de>
FCD4  DA 0D FF     jp c,FF0D        set FAC to <hl> and mark integer
FCD7  CD 4F FE     call FE4F        ...
FCDA  CD 58 BD     call BD58        REAL ARITH, ADD, (hl)=(hl)+(de)
FCDD  D8           ret c

----- Error: Overflow
FCDE  C3 F3 CA     jp CAF3          Error: Overflow

----- perform [-] (minus)
      @ CF86
FCE1  CD 15 FE     call FE15        compare VARTYPEs; change if <>; string illeg
FCE4  30 09        jr nc,FCEF       it's at least one real argument
FCE6  CD B2 BD     call BDB2        INT ARITH, SUB; <hl>=<de>-<hl>
FCE9  DA 0D FF     jp c,FF0D        set FAC to <hl> and mark integer
FCEC  CD 4F FE     call FE4F        ...
FCEF  CD 5E BD     call BD5E        REAL ARITH, SUB (hl)=(de)-(hl)
FCF2  D8           ret c
FCF3  18 E9        jr FCDE          Error: Overflow

----- perform [*] (multiply)
      @ CF8A
FCF5  CD 15 FE     call FE15        compare VARTYPEs; change if <>; string illeg
FCF8  30 09        jr nc,FD03       it's at least one real argument
FCFA  CD B5 BD     call BDB5        INT ARITH MUL; <hl>=<hl>*<de>
FCFD  DA 0D FF     jp c,FF0D        set FAC to <hl> and mark integer
FD00  CD 4F FE     call FE4F        ...
FD03  CD 61 BD     call BD61        REAL ARITH, MULT, (hl)=(hl)*(de)
FD06  D8           ret c
FD07  18 D5        jr FCDE          Error: Overflow

----- compare two numbers (int or real)
      @ C68C! CFAB! D205!
FD09  CD 15 FE     call FE15        compare VARTYPEs; change if <>; string illeg
FD0C  DA C4 BD     jp c,BDC4        INT ARITH, COMPARE <hl>,<de>; <a>= FF,00,01
FD0F  C3 6A BD     jp BD6A          REAL ARITH, COMPARE (hl),(de); <a>=FF,00,01

----- perform [/] (devide)
      @ CF8E
FD12  3A C1 B0     ld a,(B0C1)      VARTYPE
FD15  B1           or c
FD16  FE 02        cp 02            <integer VAR%>
FD18  20 05        jr nz,FD1F       it's real
FD1A  CD 4F FE     call FE4F        ...
FD1D  18 03        jr FD22
```

```
FD1F  CD 15 FE     call FE15         compare VARTYPEs; change if <>; string illeg
FD22  EB           ex de,hl
FD23  D5           push de
FD24  CD 64 BD      call BD64        REAL ARITH DVD; (hl)=(hl)/(de)
FD27  D1           pop de
FD28  F5           push af
FD29  01 05 00      ld bc,0005       # of bytes
FD2C  CD F2 FF      call FFF2        ldir
FD2F  F1           pop af
FD30  D8           ret c
FD31  CA EA CA     jp z,CAEA         Error: Division by zero
FD34  C3 F3 CA     jp CAF3           Error: Overflow

----- perform [\] (devide=integer)
      @ CF96
FD37  CD 9A FE     call FE9A         <de>=VAL(hl), <hl>=(FAC)
FD3A  EB           ex de,hl
FD3B  CD B8 BD     call BDB8         INT ARITH, DVD; <hl>=<hl>/<de>
FD3E  DA 0D FF     jp c,FF0D         set FAC to <hl> and mark integer
FD41  28 10        jr z,FD53         Error: Division by zero
FD43  21 00 80     ld hl,8000        sign bit set
FD46  C3 60 FE     jp FE60           convert unsigned integer (hl) to real

----- perform <hl> [MOD] <de>
      @ CF9E
FD49  CD 9A FE     call FE9A         <de>=VAL(hl), <hl>=(FAC)
FD4C  EB           ex de,hl
FD4D  CD BB BD     call BDBB         INT ARITH, MOD; <hl>=remainder (<hl>/<de>)
FD50  DA 0D FF     jp c,FF0D         set FAC to <hl> and mark integer

----- Error: Division by zero
FD53  1E 0B        ld e,0B           Division by zero
FD55  C3 94 CA     jp CA94           perform ERROR <e> routine

----- perform <hl> [AND] <de>
      @ CF9A
FD58  CD 9A FE     call FE9A         <de>=VAL(hl), <hl>=(FAC)
FD5B  7B           ld a,e
FD5C  A5           and 1
FD5D  6F           ld l,a
FD5E  7C           ld a,h
FD5F  A2           and d

----- set FAC to <al> and mark integer
      @ CFA6  FD6B'
FD60  C3 0C FF     jp FF0C           set FAC to <al> and mark integer

----- perform <hl> [OR] <de>
      @ CFA2
FD63  CD 9A FE     call FE9A         <de>=VAL(hl), <hl>=(FAC)
FD66  7B           ld a,e
FD67  B5           or 1
FD68  6F           ld l,a
FD69  7A           ld a,d
FD6A  B4           or h
FD6B  18 F3        jr FD60           set FAC to <al> and mark integer

----- perform <hl> [XOR] <de>
      @ CFA6
FD6D  CD 9A FE     call FE9A         <de>=VAL(hl), <hl>=(FAC)
FD70  7B           ld a,e
FD71  AD           xor 1
FD72  6F           ld l,a
FD73  7C           ld a,h
FD74  AA           xor d
```

```
FD75  18 E9       jr FD60            set FAC to <al> and mark integer

----- perform [NOT] <hl>
      @ CFC8
FD77  E5          push hl
FD78  CD 8D FE     call FE8D         function: CINT(<num expression>) in <hl>
FD7B  7D           ld a,l
FD7C  2F           cpl
FD7D  6F           ld l,a
FD7E  7C           ld a,h            complement <hl>
FD7F  2F           cpl
FD80  CD 0C FF     call FF0C         set FAC to <al> and mark integer
FD83  E1          pop hl
FD84  C9          ret

----- function: ABS(<num expression>)
      @ D1AE
FD85  CD A3 FD    call FDA3          get [SGN] <a> (FF,00,01)
FD88  F0          ret p              already positiv

      @ CFBF
FD89  E5          push hl
FD8A  C5           push bc
FD8B  CD 2D FF      call FF2D        <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FD8E  30 0D         jr nc,FD9D       it's real
FD90  CD C7 BD      call BDC7        INT ARITH, COMPLEMENT <hl>
FD93  22 C2 B0      ld (B0C2),hl     Floating point ACU, FAC
FD96  D5            push de
FD97  D4 60 FE       call nc,FE60    convert unsigned integer (hl) to real
FD9A  D1            pop de
FD9B  18 03         jr FDA0

FD9D  CD 6D BD      call BD6D        REAL ARITH, COMPLEMENT SIGN (hl)
FDA0  C1           pop bc
FDA1  E1          pop hl
FDA2  C9          ret

----- get [SGN] <a> (FF,00,01)
      @ C5BB! C6D3! C7A5! FD85! FF02!
FDA3  CD 2D FF    call FF2D          <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FDA6  DA CA BD    jp c,BDCA          INT ARITH, get SGN of <hl>; <a>= FF,00,01
FDA9  C5          push bc            proceed if real
FDAA  CD 70 BD     call BD70         REAL ARITH, SGN (hl); <a>=FF,00,01
FDAD  C1          pop bc
FDAE  C9          ret

      @ D241!
FDAF  E5          push hl
FDB0  79           ld a,c
FDB1  CD 4B FF     call FF4B         set VARTYPE <a>, copy VARIABLE (hl) to FAC
FDB4  D1          pop de
FDB5  CD 2D FF    call FF2D          <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FDB8  78          ld a,b
FDB9  30 0B       jr nc,FDC6         ...
FDBB  B7          or a
FDBC  F0          ret p
FDBD  CD 6A FE    call FE6A          convert integer (de) to real
FDC0  CD CE FD    call FDCE          ...
FDC3  C3 8D FE    jp FE8D            function: CINT(<num expression>) in <hl>

FDC6  B7          or a
FDC7  20 05       jr nz,FDCE         ...
FDC9  11 49 BD    ld de,BD49         REAL ARITH ??
FDCC  18 26       jr FDF4            ...
```

```
      @ FDC0! FDC7'
FDCE  D5          push de
FDCF  C5           push bc
FDD0  78            ld a,b
FDD1  CD 55 BD      call BD55       REAL ARITH ??
FDD4  DC 49 BD      call c,BD49     REAL ARITH ??
FDD7  78            ld a,b
FDD8  C1           pop bc
FDD9  D1          pop de
FDDA  30 08       jr nc,FDE4        ...
FDDC  CD 43 BD    call BD43         REAL ARITH, CREAL (hl) 4 byte integer to rea
FDDF  AF          xor a
FDE0  90          sub b
FDE1  C3 55 BD    jp BD55           REAL ARITH ??

      @ FDDA'
FDE4  EB          ex de,hl
FDE5  C3 4E FF    jp FF4E           copy VARIABLE (hl) to FAC

----- function: FIX(<numeric expression>)
      @ D1BE
FDE8  11 4C BD    ld de,BD4C        REAL ARITH, FIX (hl)
FDEB  18 03       jr FDF0

----- function: INT(<numeric expression>)
      @ D1C6
FDED  11 4F BD    ld de,BD4F        REAL ARITH, INT (hl)
FDF0  CD 2D FF    call FF2D         <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FDF3  D8          ret c             if already integer

      @ FDCC'
FDF4  CD FB FF    call FFFB         jp(de)
FDF7  D0          ret nc
FDF8  3A C1 B0    ld a,(B0C1)       VARTYPE
FDFB  CD 06 FE    call FE06         ...
FDFE  D8          ret c
FDFF  CD 1D FF    call FF1D         get VARTYPE <c>, <hl>=FAC
FE02  78          ld a,b
FE03  C3 43 BD    jp BD43           REAL ARITH, CREAL (hl) 4 byte integer to rea

      @ ED26! FDFB!
FE06  79          ld a,c
FE07  FE 03       cp 03             <string VAR$>
FE09  D0          ret nc            not integer, return
FE0A  7E          ld a,(hl)
FE0B  23          inc hl
FE0C  66          ld h,(hl)         ld hl,(hl)
FE0D  6F          ld l,a
FE0E  CD A9 BD    call BDA9         INT ARITH, unsigned to sign <b>; z=zero, c=+
FE11  D0          ret nc
FE12  C3 0D FF    jp FF0D           set FAC to <hl> and mark integer

----- compare VARTYPEs; change if <>; string illegal
      @ FCCC! FCE1! FCF5! FD09! FD1F!
FE15  79          ld a,c            = VARTYPE
FE16  FE 03       cp 03             <string VAR$>
FE18  28 32       jr z,FE4C         Error: Type mismatch
FE1A  3A C1 B0    ld a,(B0C1)       VARTYPE
FE1D  FE 03       cp 03             <string VAR$>
FE1F  28 2B       jr z,FE4C         Error: Type mismatch
FE21  B9          cp c              both arguments of same vartype?
FE22  28 17       jr z,FE3B         yes, both the same
FE24  30 0C       jr nc,FE32        second VARTYPE was integer
FE26  E5          push hl
FE27  21 C1 B0     ld hl,B0C1       VARTYPE
```

```
FE2A  71          ld (hl),c
FE2B  23          inc hl
FE2C  CD 63 FE    call FE63        convert signed integer (hl) to real
FE2F  D1          pop de
FE30  B7          or a
FE31  C9          ret

FE32  CD 63 FE    call FE63        convert signed integer (hl) to real
FE35  EB          ex de,hl
FE36  21 C2 B0    ld hl,B0C2       Floating point ACU, FAC
FE39  B7          or a
FE3A  C9          ret

FE3B  EE 02       xor 02           <integer VAR%>
FE3D  28 05       jr z,FE44        yes, both integer
FE3F  EB          ex de,hl
FE40  21 C2 B0    ld hl,B0C2       Floating point ACU, FAC
FE43  C9          ret              (de) points to value, (hl) to FAC

----- yes, both integer
FE44  5E          ld e,(hl)
FE45  23          inc hl
FE46  56          ld d,(hl)
FE47  2A C2 B0    ld hl,(B0C2)     Floating point ACU, FAC
FE4A  37          scf
FE4B  C9          ret              <de>=(hl); <hl>=(FAC); =carry

----- Error: Type mismatch
FE4C  C3 40 FF    jp FF40          Error: Type mismatch

      @ FCD7! FCEC! FD00! FD1A!
FE4F  2A C2 B0    ld hl,(B0C2)     Floating point ACU, FAC
FE52  CD 6A FE    call FE6A        convert integer (de) to real
FE55  2A 8B B0    ld hl,(B08B)     BASIC STACK pointer
FE58  CD 63 FE    call FE63        convert signed integer (hl) to real
FE5B  EB          ex de,hl
FE5C  21 C2 B0    ld hl,B0C2       Floating point ACU, FAC
FE5F  C9          ret

----- convert unsigned integer (hl) to real
      @ D069! D102! FC3B  FD46  FD97!
FE60  AF          xor a            =0 = unsigned integer flag
FE61  18 08       jr FE6B

----- convert signed integer (hl) to real
      @ FE2C! FE32! FE58!
FE63  5E          ld e,(hl)
FE64  23          inc hl
FE65  56          ld d,(hl)
FE66  2B          dec hl
FE67  7A          ld a,d           flag for signed/unsigned conversion
FE68  18 08       jr FE72

----- convert integer (de) to real
      @ FDBD! FE52! FEEF
FE6A  7C          ld a,h           hi byte
FE6B  EB          ex de,hl
FE6C  21 C1 B0    ld hl,B0C1       VARTYPE
FE6F  36 05       ld (hl),05       <FAC real>
FE71  23          inc hl           points to FAC
FE72  EB          ex de,hl
FE73  F5          push af
FE74  B7          or a             is it a signed conversion?
FE75  FC C7 BD    call m,BDC7      INT ARITH, COMPLEMENT <hl>
FE78  F1          pop af
```

```
FE79  C3 40 BD     jp BD40            REAL ARITH, CREAL <hl> to (de)

----- set FAC to <hl,de> and normalise to real
      @ D0E9!
FE7C  22 C2 B0     ld (B0C2),hl       Floating point ACU, FAC
FE7F  EB           ex de,hl
FE80  22 C4 B0     ld (B0C4),hl       (=FAC+2)
FE83  21 C1 B0     ld hl,B0C1         VARTYPE
FE86  36 05        ld (hl),05         <FAC real>
FE88  23           inc hl             points to FAC
FE89  AF           xor a
FE8A  C3 43 BD     jp BD43            REAL ARITH, CREAL (hl) 4 byte integer to rea

----- function: CINT(<num expression>) in <hl>
      @ C99F! CE8B! D1B6  D329! D409! D427! F16D! FA93! FD78! FDC3  FEE7'
FE8D  CD 93 FE     call FE93          perform function CINT
FE90  D8           ret c
FE91  18 3F        jr FED2            Error: Overflow

----- perform function CINT
      @ FE8D!
FE93  CD A5 FE     call FEA5          set VARTYPE integer, <a>=old VARTYPE
FE96  22 C2 B0     ld (B0C2),hl       Floating point ACU, FAC
FE99  C9           ret

----- <de>=VAL(hl), <hl>=(FAC)
      @ FD37! FD49! FD58! FD63! FD6D!
FE9A  79           ld a,c             = VARTYPE
FE9B  CD AC FE     call FEAC          <hl>=integer VAL(hl)
FE9E  EB           ex de,hl
FE9F  DC A5 FE     call c,FEA5        set VARTYPE integer, <a>=old VARTYPE
FEA2  D8           ret c
FEA3  18 2D        jr FED2            Error: Overflow

----- set VARTYPE integer, <a>=old VARTYPE
      @ FE93! FE9F!
FEA5  21 C1 B0     ld hl,B0C1         VARTYPE
FEA8  7E           ld a,(hl)
FEA9  36 02        ld (hl),02         set integer
FEAB  23           inc hl             hl points to FAC now

----- <hl>=integer VAL(hl)
      @ FE9B!
FEAC  FE 03        cp 03              <string VAR$>
FEAE  38 0D        jr c,FEBD          it is INTEGER; ld hl,(hl)
FEB0  CA 40 FF     jp z,FF40          Error: Type mismatch
FEB3  C5           push bc
FEB4  CD 46 BD      call BD46         REAL ARITH, CINT; <hl>=int(hl); <a>=sign
FEB7  47            ld b,a
FEB8  DC A9 BD      call c,BDA9       INT ARITH, unsigned to sign <b>; z=zero, c=+
FEBB  C1           pop bc
FEBC  C9           ret

----- it is INTEGER; ld hl,(hl)
FEBD  7E           ld a,(hl)
FEBE  23           inc hl
FEBF  66           ld h,(hl)
FEC0  6F           ld l,a
FEC1  C9           ret

----- function: UNT(<address expression>)
      @ CE97! D1E4  F124! F158! FCC3!
FEC2  CD 2D FF     call FF2D          <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FEC5  D8           ret c              already integer
FEC6  CD 46 BD     call BD46          REAL ARITH, CINT; <hl>=int(hl); <a>=sign
```

```
FEC9  30 07       jr nc,FED2        Error: Overflow
FECB  47          ld b,a
FECC  FC A9 BD    call m,BDA9       INT ARITH, unsigned to sign <b>; z=zero, c=+
FECF  DA 0D FF    jp c,FF0D         set FAC to <hl> and mark integer

----- Error: Overflow
FED2  1E 06       ld e,06           Overflow
FED4  C3 94 CA    jp CA94           perform ERROR <e> routine

----- test <a>=VARTYPE? if not CINT,CREAL
      @ C584! C598! C5B4! D18D  D66C!
FED7  E5          push hl
FED8  D5           push de
FED9  C5            push bc
FEDA  21 C1 B0       ld hl,B0C1     VARTYPE
FEDD  BE             cp (hl)        <a>=VARTYPE?
FEDE  C4 E5 FE       call nz,FEE5   goto (CINT CREAL or STRING), <a>
FEE1  C1            pop bc
FEE2  D1           pop de
FEE3  E1          pop hl
FEE4  C9          ret

----- goto (CINT CREAL or STRING), <a>
      @ FEDE!
FEE5  D6 03       sub 03            <string VAR$>
FEE7  38 A4       jr c,FE8D         function: CINT(<num expression>) in <hl>
FEE9  CA 3C FF    jp z,FF3C         test VARTYPE for string, else error

----- function: CREAL(<numeric expression>)
      @ D1BA  D4F6! D51B! D57C! D594!
FEEC  CD 2D FF    call FF2D         <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FA
FEEF  DA 6A FE    jp c,FE6A         convert integer (de) to real
FEF2  C9          ret

----- set FAC to all zeroes
      @ D01F  ECB9!
FEF3  E5          push hl
FEF4  21 00 00     ld hl,0000
FEF7  22 C2 B0     ld (B0C2),hl     Floating point ACU, FAC
FEFA  22 C4 B0     ld (B0C4),hl
FEFD  22 C5 B0     ld (B0C5),hl
FF00  E1          pop hl
FF01  C9          ret

----- function: SGN(<numeric expression>)
      @ D1D6
FF02  CD A3 FD    call FDA3         get [SGN] <a> (FF,00,01)

----- set FAC to (-1, 0, +1); <a> was FF,00,01
      @ C41F! CFB6
FF05  6F          ld l,a
FF06  87          add a,a
FF07  9F          sbc a,a
FF08  18 02       jr FF0C           set FAC to <al> and mark integer

----- set FAC to <a> and mark integer
      @ C288! C500! D0E0! D33E  D433  F15C  F174  FA0D  FA13  FA7A  FACF!
FF0A  6F          ld l,a
FF0B  AF          xor a

----- set FAC to <al> and mark integer
      @ FD60  FD80! FF08'
FF0C  67          ld h,a
```

```
----- set FAC to <hl> and mark integer
      @ C5A4! C9AE  D053! D112! D420  E170! E283! E29B! ECD2! EE0A! EE79!
      @ FCD4  FCE9  FCFD  FD3E  FD50  FE12  FECF
FF0D  22 C2 B0     ld (B0C2),hl      Floating point ACU, FAC

----- set VARTYPE integer; <a>=2
      @ ECB6! ECF0!
FF10  3E 02        ld a,02           <integer VAR%>
FF12  32 C1 B0     ld (B0C1),a       VARTYPE
FF15  C9           ret

----- ld hl,FAC; set VARTYPE real; <a>=5
      @ D5A6! ED37!
FF16  21 C2 B0     ld hl,B0C2        Floating point ACU, FAC

----- set VARTYPE real; <a>=5
      @ D4DC! ED04! ED90!
FF19  3E 05        ld a,05           real
FF1B  18 F5        jr FF12           set VARTYPE

----- get VARTYPE <c>, <hl>=FAC
      @ D4DF! FDFF!
FF1D  21 C1 B0     ld hl,B0C1        VARTYPE
FF20  4E           ld c,(hl)
FF21  23           inc hl
FF22  C9           ret

----- get VARTYPE <a>;
      @ D667! E071!
FF23  3A C1 B0     ld a,(B0C1)       VARTYPE
FF26  C9           ret

----- get VARTYPE <a>; cp string
      @ E022! ED1F!
FF27  3A C1 B0     ld a,(B0C1)       VARTYPE
FF2A  FE 03        cp 03             <string VAR$>
FF2C  C9           ret

----- <a>=VARTYPE;3=err;ld hl,(FAC);ret c;ld hl,FAC
      @ FCB3! FD8B! FDA3! FDB5! FDF0! FEC2! FEEC!
FF2D  3A C1 B0     ld a,(B0C1)       VARTYPE
FF30  FE 03        cp 03             <string VAR$>
FF32  28 0C        jr z,FF40         Error: Type mismatch
FF34  2A C2 B0     ld hl,(B0C2)      Floating point ACU, FAC
FF37  D8           ret c
FF38  21 C2 B0     ld hl,B0C2        Floating point ACU, FAC
FF3B  C9           ret

----- test VARTYPE for string, else error
      @ CEA8  CF27! DB06! F382! F99A! FBDB! FEE9
FF3C  CD 45 FF     call FF45         get VARTYPE <a>; cp string
FF3F  C8           ret z

----- Error: Type mismatch
FF40  1E 0D        ld e,0D           Type mismatch
FF42  C3 94 CA     jp CA94           perform ERROR <e> routine

----- get VARTYPE <a>; cp string
      @ CF19! CF62! D185! D66F! DC03! F238! F489! FA46! FAA4! FC2D! FF3C!
FF45  3A C1 B0     ld a,(B0C1)       VARTYPE
FF48  FE 03        cp 03             is it a string?
FF4A  C9           ret
```

```
----- set VARTYPE <a>, copy VARIABLE (hl) to FAC
      @ C66F! D018! D043! D503! E2CA! F11E! F906! FDB1!
FF4B  32 C1 B0     ld (B0C1),a        VARTYPE

----- copy VARIABLE (hl) to FAC
      @ D20E! FDE5
FF4E  11 C2 B0     ld de,B0C2         Floating point ACU, FAC
FF51  18 13        jr FF66            copy variable (hl) to (de)

----- get VARTYPE, copy FAC to BASIC STACK
      @ CF42! D1F8! D21C! F8D1! F8F5!
FF53  D5           push de
FF54  E5            push hl
FF55  3A C1 B0       ld a,(B0C1)      VARTYPE
FF58  4F             ld c,a           number of bytes
FF59  CD B0 F5       call F5B0        inc BASIC STACK pointer by <a>, (hl)=next lo
FF5C  CD 62 FF       call FF62        copy FAC to (hl)
FF5F  E1            pop hl
FF60  D1           pop de
FF61  C9           ret

----- copy FAC to (hl)
      @ C58B! C59B! C5B8! C685! D672  FF5C!
FF62  EB           ex de,hl
FF63  21 C2 B0     ld hl,B0C2         Floating point ACU, FAC

----- copy variable (hl) to (de)
      @ C5E6! D67A  FF51'
FF66  C5           push bc
FF67  3A C1 B0      ld a,(B0C1)       VARTYPE
FF6A  4F            ld c,a
FF6B  06 00         ld b,00
FF6D  ED B0         ldir
FF6F  C1           pop bc
FF70  C9           ret

----- test <a> for A-Z, =carry
      @ D61F! D630! DEE7! FF7B!
FF71  CD 8A FF     call FF8A          change <a> to upper case
FF74  FE 41        cp 41              'A
FF76  3F           ccf
FF77  D0           ret nc
FF78  FE 5B        cp 5B              '[
FF7A  C9           ret

----- test <a> for A-Z, =carry
      @ DF64! DF6F! DF90! DFB1! E0D9! E243!
FF7B  CD 71 FF     call FF71          test <a> for A-Z, =carry
FF7E  D8           ret c

----- test <a> for '.' or digit, =CARRY
      @ DEEC! ECB1!
FF7F  FE 2E        cp 2E              '.
FF81  37           scf
FF82  C8           ret z

----- test <a> for digit, =carry
      @ ECE4! ED58! ED85! EE63!
FF83  FE 30        cp 30              '0
FF85  3F           ccf
FF86  D0           ret nc
FF87  FE 3A        cp 3A              ':
FF89  C9           ret
```

```
----- change <a> to upper case
      @ DF56! E33D! ED5E  EE20! EE68! F842: FF71!
FF8A  FE 61        cp 61             'a
FF8C  D8           ret c
FF8D  FE 7B        cp 7B             '{
FF8F  D0           ret nc
FF90  D6 20        sub 20            'SPACE
FF92  C9           ret

----- search <a> in table(hl); hl=address
      @ CFE6! EC27! F217!
FF93  F5           push af
FF94  C5            push bc
FF95  46             ld b,(hl)
FF96  23             inc hl
FF97  E5             push hl
FF98  23              inc hl         skip over address part
FF99  23              inc hl
FF9A  BE              cp (hl)
FF9B  23              inc hl
FF9C  28 04           jr z,FFA2      match found
FF9E  05              dec b
FF9F  20 F7           jr nz,FF98     next entry
FFA1  E3              ex (sp),hl
FFA2  F1             pop af
FFA3  7E             ld a,(hl)
FFA4  23             inc hl
FFA5  66             ld h,(hl)
FFA6  6F             ld l,a
FFA7  C1            pop bc
FFA8  F1           pop af
FFA9  C9           ret

----- search <a> in table(hl); =carry
      @ DF16! DFCD!
FFAA  C5           push bc
FFAB  4F            ld c,a
FFAC  7E            ld a,(hl)
FFAD  B7            or a
FFAE  28 05         jr z,FFB5        end of table
FFB0  23            inc hl
FFB1  B9            cp c
FFB2  20 F8         jr nz,FFAC       get next byte
FFB4  37            scf
FFB5  79            ld a,c
FFB6  C1           pop bc
FFB7  C9           ret

----- test HL=DE? (try hl-de)
      @ C54E! C654! C7D9! D235! D40F! D838! D9A3! E053! E12A! E778! E7B6!
      @ E7D3! E814! EADF! EAF9! F512! F5EE! F625  F6D7! F732! F757! F766!
      @ FB4F! FB68! FB70! FB78! FB82! FBC8! FBF1! FC0E! FC82!
FFB8  7C           ld a,h
FFB9  92           sub d
FFBA  C0           ret nz
FFBB  7D           ld a,l
FFBC  93           sub e
FFBD  C9           ret

----- test HL=BC? (try hl-bc)
      @ DABE! EBCE! F4C7! F74A! F7BE! FB31! FB39! FC9F! FCA6!
FFBE  7C           ld a,h
FFBF  90           sub b
FFC0  C0           ret nz
FFC1  7D           ld a,l
```

```
FFC2  91           sub c
FFC3  C9           ret

----- ED=HL-ED
FFC4  C5           push bc            good old 8080
FFC5  47            ld b,a
FFC6  7D            ld a,l
FFC7  93            sub e
FFC8  5F            ld e,a
FFC9  7C            ld a,h
FFCA  9A            sbc a,d
FFCB  57            ld d,a
FFCC  78            ld a,b
FFCD  C1           pop bc
FFCE  C9           ret                never used

----- HL=HL-DE
      @ D9E9! E712! E749! EBCB! EC52! F509! F50D! F550! F62F  F747! F76F!
FFCF  C5           push bc
FFD0  47            ld b,a
FFD1  7D            ld a,l
FFD2  93            sub e
FFD3  6F            ld l,a
FFD4  7C            ld a,h
FFD5  9A            sbc a,d
FFD6  67            ld h,a
FFD7  78            ld a,b
FFD8  C1           pop bc
FFD9  C9           ret

----- BC=HL-DE
      @ C145! E722! EAC7! F526! F557!
FFDA  E5           push hl
FFDB  67            ld h,a
FFDC  E3            ex (sp),hl
FFDD  7D            ld a,l
FFDE  93            sub e
FFDF  4F            ld c,a
FFE0  7C            ld a,h
FFE1  9A            sbc a,d
FFE2  47            ld b,a
FFE3  E3            ex (sp),hl
FFE4  7C            ld a,h
FFE5  E1           pop hl
FFE6  C9           ret

----- HL=HL-BC
FFE7  D5           push de            good old 8080
FFE8  57            ld d,a
FFE9  7D            ld a,l
FFEA  91            sub c
FFEB  6F            ld l,a
FFEC  7C            ld a,h
FFED  98            sbc a,b
FFEE  67            ld h,a
FFEF  7A            ld a,d
FFF0  D1           pop de
FFF1  C9           ret                never used

----- ldir
      @ D797! D9F1! E704  E71B! EB53! F582! F7A8! F891! F933! F98B! F9E4!
      @ FB9E! FD2C!
FFF2  ED B0        ldir
FFF4  C9           ret
```

```
----- lddr
      @ EAD0! F565! F611! F793! FC6B!
FFF5  ED B8        lddr
FFF7  C9           ret

----- jp(hl)
      @ D48F! DAF1!
FFF8  E9           jp (hl)

----- jp(bc)
      @ C1DC! C21C! C285! C4FD! C515! D4BE! E912! F85B!
FFF9  C5           push bc
FFFA  C9            ret

----- jp(de)
      @ CF54! D3E2! DC74! F21B! F7FD! FDF4!
FFFB  D5           push de
FFFC  C9            ret

-----
FFFD  C7 C7 54
```

## Nobody is perfect!

Werden bei der Eingabe einer Basic-Zeile am Schluß noch Blanks angehängt oder, was viel öfter passiert, wird eine Basic-Zeile mittels Copy-Cursor übernommen - und wer stoppt schon genau hinter dem letzten Zeichen -, so werden alle überflüssigen Blanks mit ins Basic-Programm eingebaut. Wer das nicht weiß, verschwendet bald erheblichen Speicherplatz. Es ist eine Routine vorhanden, die bei der Zeilen-Assemblierung alle überflüssigen Blanks, Tabs und CR's eliminiert und ein Flag (AC00), das den Rechner dazu veranlaßt. Geben Sie also vor einer Programmierung im Direktmodus ein:

POKE &AC00,255

Sie werden sich bald an die 'enge' Schreibweise gewöhnen. Leider funktioniert das nur bei der Direkteingabe oder bei der Übernahme einer Zeile mit dem Copy-Cursor. Bestehende Programme lassen sich leider nicht mehr so elegant komprimieren.

| HEX | DEC | ASCII | TOKEN | Z80 CODE | CB XX | ED XX | HEX |
|---|---|---|---|---|---|---|---|
| 00 | 0 | 'NUL | [ABS] | nop | rlc b | | 00 |
| 01 | 1 | 'SOH (^A) | [ATN] | ld bc,**** | rlc c | | 01 |
| 02 | 2 | 'STX (^B) | [AUTO] | ld (bc),a | rlc d | | 02 |
| 03 | 3 | 'ETX (^C) | [CINT] | inc bc | rlc e | | 03 |
| 04 | 4 | 'EOT (^D) | [CLEAR] | inc b | rlc h | | 04 |
| 05 | 5 | 'ENQ (^E) | [COS] | dec b | rlc l | | 05 |
| 06 | 6 | 'ACK (^F) | [CREAL] | ld b,** | rlc (hl) | | 06 |
| 07 | 7 | 'BEL (^G) | [EXP] | rlca | rlc a | | 07 |
| 08 | 8 | 'BS (^H) | [FIX] | ex af,af' | rrc b | | 08 |
| 09 | 9 | 'HT (^I) | [FRE] | add hl,bc | rrc c | | 09 |
| 0A | 10 | 'LF (^J) | [INKEY] | ld a,(bc) | rrc d | | 0A |
| 0B | 11 | 'VT (^K) | [INP] | dec bc | rrc e | | 0B |
| 0C | 12 | 'FF (^L) | [INT] | inc c | rrc h | | 0C |
| 0D | 13 | 'CR (^M) | [JOY] | dec c | rrc l | | 0D |
| 0E | 14 | 'SO (^N) | [LEN] | ld c,** | rrc (hl) | | 0E |
| 0F | 15 | 'SI (^O) | [LOG] | rrca | rrc a | | 0F |
| 10 | 16 | 'DLE (^P) | [LOG10] | djnz ** | rl b | | 10 |
| 11 | 17 | 'DC1 (^Q) | [LOWER$] | ld de,**** | rl c | | 11 |
| 12 | 18 | 'DC2 (^R) | [PEEK] | ld (de),a | rl d | | 12 |
| 13 | 19 | 'DC3 (^S) | [REMAIN] | inc de | rl e | | 13 |
| 14 | 20 | 'DC4 (^T) | [SGN] | inc d | rl h | | 14 |
| 15 | 21 | 'NAK (^U) | [SIN] | dec d | rl l | | 15 |
| 16 | 22 | 'SYN (^V) | [SPACE$] | ld d,** | rl (hl) | | 16 |
| 17 | 23 | 'ETB (^W) | [SQ] | rla | rl a | | 17 |
| 18 | 24 | 'CAN (^X) | [SQR] | jr ** | rr b | | 18 |
| 19 | 25 | 'EM (^Y) | [STR$] | add hl,de | rr c | | 19 |
| 1A | 26 | 'SUB (^Z) | [TAN] | ld a,(de) | rr d | | 1A |
| 1B | 27 | 'ESC | [UNT] | dec de | rr e | | 1B |
| 1C | 28 | 'FS | [UPPER$] | inc e | rr h | | 1C |
| 1D | 29 | 'GS | | dec e | rr l | | 1D |
| 1E | 30 | 'RS | [VAL] | ld e,** | rr (hl) | | 1E |
| 1F | 31 | 'US | | rra | rr a | | 1F |
| 20 | 32 | 'SPACE | | jr nz,** | sla b | | 20 |
| 21 | 33 | '! | | ld hl,**** | sla c | | 21 |
| 22 | 34 | '" | | ld (****),hl | sla d | | 22 |
| 23 | 35 | '# | | inc hl | sla e | | 23 |
| 24 | 36 | '$ | | inc h | sla h | | 24 |
| 25 | 37 | '% | | dec h | sla l | | 25 |
| 26 | 38 | '& | | ld h,** | sla (hl) | | 26 |
| 27 | 39 | '' | | daa | sla a | | 27 |
| 28 | 40 | '( | | jr z,** | sra b | | 28 |
| 29 | 41 | ') | | add hl,hl | sra c | | 29 |
| 2A | 42 | '* | | ld hl,(****) | sra d | | 2A |
| 2B | 43 | '+ | | dec hl | sra e | | 2B |
| 2C | 44 | ', | | inc l | sra h | | 2C |
| 2D | 45 | '- | | dec l | sra l | | 2D |
| 2E | 46 | '. | | ld l,** | sra (hl) | | 2E |
| 2F | 47 | '/ | | cpl | sra a | | 2F |
| 30 | 48 | '0 | | jr nc,** | | | 30 |
| 31 | 49 | '1 | | ld sp,**** | | | 31 |
| 32 | 50 | '2 | | ld (****),a | | | 32 |
| 33 | 51 | '3 | | inc sp | | | 33 |
| 34 | 52 | '4 | | inc (hl) | | | 34 |
| 35 | 53 | '5 | | dec (hl) | | | 35 |
| 36 | 54 | '6 | | ld (hl),** | | | 36 |
| 37 | 55 | '7 | | scf | | | 37 |
| 38 | 56 | '8 | | jr c,** | srl b | | 38 |
| 39 | 57 | '9 | | add hl,sp | srl c | | 39 |
| 3A | 58 | ': | | ld a,(****) | srl d | | 3A |
| 3B | 59 | '; | | dec sp | srl e | | 3B |
| 3C | 60 | '< | | inc a | srl h | | 3C |
| 3D | 61 | '= | | dec a | srl l | | 3D |
| 3E | 62 | '> | | ld a,** | srl (hl) | | 3E |
| 3F | 63 | '? | | ccf | srl a | | 3F |

| HEX | DEC | ASCII | TOKEN | Z80 CODE | CB XX | ED XX | HEX |
|---|---|---|---|---|---|---|---|
| 40 | 64 | '@ | [EOF] | ld b,b | bit 0,b | in b,(c) | 40 |
| 41 | 65 | 'A | [ERR] | ld b,c | bit 0,c | out (c),b | 41 |
| 42 | 66 | 'B | [HIMEM] | ld b,d | bit 0,d | sbc hl,bc | 42 |
| 43 | 67 | 'C | [INKEY$] | ld b,e | bit 0,e | ld (****),bc | 43 |
| 44 | 68 | 'D | [RND] | ld b,h | bit 0,h | neg | 44 |
| 45 | 69 | 'E | [TIME] | ld b,l | bit 0,l | retn | 45 |
| 46 | 70 | 'F | [XPOS] | ld b,(hl) | bit 0,(hl) | im 0 | 46 |
| 47 | 71 | 'G | [YPOS] | ld b,a | bit 0,a | ld i,a | 47 |
| 48 | 72 | 'H | | ld c,b | bit 1,b | in c,(c) | 48 |
| 49 | 73 | 'I | | ld c,c | bit 1,c | out (c),c | 49 |
| 4A | 74 | 'J | | ld c,d | bit 1,d | adc hl,bc | 4A |
| 4B | 75 | 'K | | ld c,e | bit 1,e | ld bc,(****) | 4B |
| 4C | 76 | 'L | | ld c,h | bit 1,h | | 4C |
| 4D | 77 | 'M | | ld c,l | bit 1,l | reti | 4D |
| 4E | 78 | 'N | | ld c,(hl) | bit 1,(hl) | | 4E |
| 4F | 79 | 'O | | ld c,a | bit 1,a | ld r,a | 4F |
| 50 | 80 | 'P | | ld d,b | bit 2,b | in d,(c) | 50 |
| 51 | 81 | 'Q | | ld d,c | bit 2,c | out (c),d | 51 |
| 52 | 82 | 'R | | ld d,d | bit 2,d | sbc hl,de | 52 |
| 53 | 83 | 'S | | ld d,e | bit 2,e | ld (****),de | 53 |
| 54 | 84 | 'T | | ld d,h | bit 2,h | | 54 |
| 55 | 85 | 'U | | ld d,l | bit 2,l | | 55 |
| 56 | 86 | 'V | | ld d,(hl) | bit 2,(hl) | im 1 | 56 |
| 57 | 87 | 'W | | ld d,a | bit 2,a | ld a,i | 57 |
| 58 | 88 | 'X | | ld e,b | bit 2,b | in e,(c) | 58 |
| 59 | 89 | 'Y | | ld e,c | bit 3,c | out (c),e | 59 |
| 5A | 90 | 'Z | | ld e,d | bit 3,d | adc hl,de | 5A |
| 5B | 91 | '[ | | ld e,e | bit 3,e | ld de,(****) | 5B |
| 5C | 92 | '\ | | ld e,h | bit 3,h | | 5C |
| 5D | 93 | '] | | ld e,l | bit 3,l | | 5D |
| 5E | 94 | '^ | | ld e,(hl) | bit 3,(hl) | im 2 | 5E |
| 5F | 95 | '_ | | ld e,a | bit 3,a | ld a,r | 5F |
| 60 | 96 | '` | | ld h,b | bit 4,b | in h,(c) | 60 |
| 61 | 97 | 'a | | ld h,c | bit 4,c | out (c),h | 61 |
| 62 | 98 | 'b | | ld h,d | bit 4,d | sbc hl,hl | 62 |
| 63 | 99 | 'c | | ld h,e | bit 4,e | ld (****),hl | 63 |
| 64 | 100 | 'd | | ld h,h | bit 4,h | | 64 |
| 65 | 101 | 'e | | ld h,l | bit 4,l | | 65 |
| 66 | 102 | 'f | | ld h,(hl) | bit 4,(hl) | | 66 |
| 67 | 103 | 'g | | ld h,a | bit 4,a | rrd | 67 |
| 68 | 104 | 'h | | ld l,b | bit 5,b | in l,(c) | 68 |
| 69 | 105 | 'i | | ld l,c | bit 5,c | out (c),l | 69 |
| 6A | 106 | 'j | | ld l,d | bit 5,d | adc hl,hl | 6A |
| 6B | 107 | 'k | | ld l,e | bit 5,e | ld hl,(****) | 6B |
| 6C | 108 | 'l | | ld l,h | bit 5,h | | 6C |
| 6D | 109 | 'm | | ld l,l | bit 5,l | | 6D |
| 6E | 110 | 'n | | ld l,(hl) | bit 5,(hl) | | 6E |
| 6F | 111 | 'o | | ld l,a | bit 5,a | rld | 6F |
| 70 | 112 | 'p | | ld (hl),b | bit 6,b | | 70 |
| 71 | 113 | 'q | [BORDER] | ld (hl),c | bit 6,c | | 71 |
| 72 | 114 | 'r | [DEC$] | ld (hl),d | bit 6,d | sbc hl,sp | 72 |
| 73 | 115 | 's | [HEX$] | ld (hl),e | bit 6,e | ld (****),sp | 73 |
| 74 | 116 | 't | [INSTR] | ld (hl),h | bit 6,h | | 74 |
| 75 | 117 | 'u | [LEFT$] | ld (hl),l | bit 6,l | | 75 |
| 76 | 118 | 'v | [MAX] | halt | bit 6,(hl) | | 76 |
| 77 | 119 | 'w | [MIN] | ld (hl),a | bit 6,a | | 77 |
| 78 | 120 | 'x | [POS] | ld a,b | bit 7,b | in a,(c) | 78 |
| 79 | 121 | 'y | [RIGHT$] | ld a,c | bit 7,c | out (c),a | 79 |
| 7A | 122 | 'z | [ROUND] | ld a,d | bit 7,d | adc hl,sp | 7A |
| 7B | 123 | '{ | [STRING$] | ld a,e | bit 7,e | ld sp,(****) | 7B |
| 7C | 124 | '\| | [TEST] | ld a,h | bit 7,h | | 7C |
| 7D | 125 | '} | [TESTR] | ld a,l | bit 7,l | | 7D |
| 7E | 126 | '~ | | ld a,(hl) | bit 7,(hl) | | 7E |
| 7F | 127 | 'DEL | [VPOS] | ld a,a | bit 7,a | | 7F |

| HEX | DEC | ASCII | TOKEN | Z80 CODE | CB XX | ED XX | HEX |
|---|---|---|---|---|---|---|---|
| 80 | 128 | 'F0 0 | [AFTER] | add a,b | res 0,b | | 80 |
| 81 | 129 | 'F1 1 | [AUTO] | add a,c | res 0,c | | 81 |
| 82 | 130 | 'F2 2 | [BORDER] | add a,d | res 0,d | | 82 |
| 83 | 131 | 'F3 3 | [CALL] | add a,e | res 0,e | | 83 |
| 84 | 132 | 'F4 4 | [CAT] | add a,h | res 0,h | | 84 |
| 85 | 133 | 'F5 5 | [CHAIN] | add a,l | res 0,l | | 85 |
| 86 | 134 | 'F6 6 | [CLEAR] | add a,(hl) | res 0,(hl) | | 86 |
| 87 | 135 | 'F7 7 | [CLG] | add a,a | res 0,a | | 87 |
| 88 | 136 | 'F8 8 | [CLOSEIN] | adc a,b | res 1,b | | 88 |
| 89 | 137 | 'F9 9 | [CLOSEOUT] | adc a,c | res 1,c | | 89 |
| 8A | 138 | 'F10 . | [CLS] | adc a,d | res 1,d | | 8A |
| 8B | 139 | 'F11 <CR> | [CONT] | adc a,e | res 1,e | | 8B |
| 8C | 140 | 'F12 RUN"<CR> | [DATA] | adc a,h | res 1,h | | 8C |
| 8D | 141 | 'F13 | [DEF] | adc a,l | res 1,l | | 8D |
| 8E | 142 | 'F14 | [DEFINT] | adc a,(hl) | res 1,(hl) | | 8E |
| 8F | 143 | 'F15 | [DEFREAL] | adc a,a | res 1,a | | 8F |
| 90 | 144 | 'F16 | [DEFSTR] | sub b | res 2,b | | 90 |
| 91 | 145 | 'F17 | [DEG] | sub c | res 2,c | | 91 |
| 92 | 146 | 'F18 | [DELETE] | sub d | res 2,d | | 92 |
| 93 | 147 | 'F19 | [DIM] | sub e | res 2,e | | 93 |
| 94 | 148 | 'F20 | [DRAW] | sub h | res 2,h | | 94 |
| 95 | 149 | 'F21 | [DRAWR] | sub l | res 2,l | | 95 |
| 96 | 150 | 'F22 | [EDIT] | sub (hl) | res 2,(hl) | | 96 |
| 97 | 151 | 'F23 | [ELSE] | sub a | res 2,a | | 97 |
| 98 | 152 | 'F24 | [END] | sbc a,b | res 3,b | | 98 |
| 99 | 153 | 'F25 | [ENT] | sbc a,c | res 3,c | | 99 |
| 9A | 154 | 'F26 | [ENV] | sbc a,d | res 3,d | | 9A |
| 9B | 155 | 'F27 | [ERASE] | sbc a,e | res 3,e | | 9B |
| 9C | 156 | 'F28 | [ERROR] | sbc a,h | res 3,h | | 9C |
| 9D | 157 | 'F29 | [EVERY] | sbc a,l | res 3,l | | 9D |
| 9E | 158 | 'F30 | [FOR] | sbc a,(hl) | res 3,(hl) | | 9E |
| 9F | 159 | 'F31 | [GOSUB] | sbc a,a | res 3,a | | 9F |
| A0 | 160 | | [GOTO] | and b | res 4,b | ldi | A0 |
| A1 | 161 | | [IF] | and c | res 4,c | cpi | A1 |
| A2 | 162 | | [INK] | and d | res 4,d | ini | A2 |
| A3 | 163 | 'POUND | [INPUT] | and e | res 4,e | outi | A3 |
| A4 | 164 | | [KEY] | and h | res 4,h | | A4 |
| A5 | 165 | | [LET] | and l | res 4,l | | A5 |
| A6 | 166 | | [LINE] | and (hl) | res 4,(hl) | | A6 |
| A7 | 167 | | [LIST] | and a | res 4,a | | A7 |
| A8 | 168 | | [LOAD] | xor b | res 5,b | ldd | A8 |
| A9 | 169 | | [LOCATE] | xor c | res 5,c | cpd | A9 |
| AA | 170 | | [MEMORY] | xor d | res 5,d | ind | AA |
| AB | 171 | | [MERGE] | xor e | res 5,e | outd | AB |
| AC | 172 | | [MID$] | xor h | res 5,h | | AC |
| AD | 173 | | [MODE] | xor l | res 5,l | | AD |
| AE | 174 | | [MOVE] | xor (hl) | res 5,(hl) | | AE |
| AF | 175 | | [MOVER] | xor a | res 5,a | | AF |
| B0 | 176 | | [NEXT] | or b | res 6,b | ldir | B0 |
| B1 | 177 | | [NEW] | or c | res 6,c | cpir | B1 |
| B2 | 178 | | [ON] | or d | res 6,d | inir | B2 |
| B3 | 179 | | [ON BREAK] | or e | res 6,e | otir | B3 |
| B4 | 180 | | [ON ERROR] | or h | res 6,h | | B4 |
| B5 | 181 | | [ON SQ] | or l | res 6,l | | B5 |
| B6 | 182 | | [OPENIN] | or (hl) | res 6,(hl) | | B6 |
| B7 | 183 | | [OPENOUT] | or a | res 6,a | | B7 |
| B8 | 184 | | [ORIGIN] | cp b | res 7,b | lddr | B8 |
| B9 | 185 | | [OUT] | cp c | res 7,c | cpdr | B9 |
| BA | 186 | | [PAPER] | cp d | res 7,d | indr | BA |
| BB | 187 | | [PEN] | cp e | res 7,e | otdr | BB |
| BC | 188 | | [PLOT] | cp h | res 7,h | | BC |
| BD | 189 | | [PLOTR] | cp l | res 7,l | | BD |
| BE | 190 | | [POKE] | cp (hl) | res 7,(hl) | | BE |
| BF | 191 | | [PRINT] | cp a | res 7,a | | BF |

| HEX | DEC | ASCII | TOKEN | Z80 CODE | CB XX | ED XX | HEX |
|---|---|---|---|---|---|---|---|
| C0 | 192 | | [REM'] | ret nz | set 0,b | | C0 |
| C1 | 193 | | [RAD] | pop bc | set 0,c | | C1 |
| C2 | 194 | | [RANDOMIZE] | jp nz,**** | set 0,d | | C2 |
| C3 | 195 | | [READ] | jp **** | set 0,e | | C3 |
| C4 | 196 | | [RELEASE] | call nz,**** | set 0,h | | C4 |
| C5 | 197 | | [REM] | push bc | set 0,l | | C5 |
| C6 | 198 | | [RENUM] | add a,** | set 0,(hl) | | C6 |
| C7 | 199 | | [RESTORE] | rst 0 | set 0,a | | C7 |
| C8 | 200 | | [RESUME] | ret z | set 1,b | | C8 |
| C9 | 201 | | [RETURN] | ret | set 1,c | | C9 |
| CA | 202 | | [RUN] | jp z,**** | set 1,d | | CA |
| CB | 203 | | [SAVE] | | set 1,e | | CB |
| CC | 204 | | [SOUND] | call z,**** | set 1,h | | CC |
| CD | 205 | | [SPEED] | call **** | set 1,l | | CD |
| CE | 206 | | [STOP] | adc a,** | set 1,(hl) | | CE |
| CF | 207 | | [SYMBOL] | rst 1,**** | set 1,a | | CF |
| D0 | 208 | | [TAG] | ret nc | set 2,b | | D0 |
| D1 | 209 | | [TAGOFF] | pop de | set 2,c | | D1 |
| D2 | 210 | | [TROFF] | jp nc,**** | set 2,d | | D2 |
| D3 | 211 | | [TRON] | out **,a | set 2,e | | D3 |
| D4 | 212 | | [WAIT] | call nc,**** | set 2,h | | D4 |
| D5 | 213 | | [WEND] | push de | set 2,l | | D5 |
| D6 | 214 | | [WHILE] | sub ** | set 2,(hl) | | D6 |
| D7 | 215 | | [WIDTH] | rst 2,**** | set 2,a | | D7 |
| D8 | 216 | | [WINDOW] | ret c | set 3,b | | D8 |
| D9 | 217 | | [WRITE] | exx | set 3,c | | D9 |
| DA | 218 | | [ZONE] | jp c,**** | set 3,d | | DA |
| DB | 219 | | [DI] | in a,** | set 3,e | | DB |
| DC | 220 | | [EI] | call c,**** | set 3,h | | DC |
| DD | 221 | | | | set 3,l | | DD |
| DE | 222 | | | sbc a,** | set 3,(hl) | | DE |
| DF | 223 | | | rst 3,**** | set 3,a | | DF |
| E0 | 224 | 'COPY KEY | | ret po | set 4,b | | E0 |
| E1 | 225 | 'INS (^TAB) | | pop hl | set 4,c | | E1 |
| E2 | 226 | | | jp po,**** | set 4,d | | E2 |
| E3 | 227 | | [ERL] | ex (sp),hl | set 4,e | | E3 |
| E4 | 228 | | [FN] | call po,**** | set 4,h | | E4 |
| E5 | 229 | | [SPC] | push hl | set 4,l | | E5 |
| E6 | 230 | | [STEP] | and ** | set 4,(hl) | | E6 |
| E7 | 231 | | [SWAP] | rst 4 | set 4,a | | E7 |
| E8 | 232 | | | ret pe | set 5,b | | E8 |
| E9 | 233 | | | jp (hl) | set 5,c | | E9 |
| EA | 234 | | [TAB] | jp pe,**** | set 5,d | | EA |
| EB | 235 | | [THEN] | ex de,hl | set 5,e | | EB |
| EC | 236 | | [TO] | call pe,**** | set 5,h | | EC |
| ED | 237 | | [USING] | | set 5,l | | ED |
| EE | 238 | | [>] | xor ** | set 5,(hl) | | EE |
| EF | 239 | 'BREAK mark | [=] | rst 5,**** | set 5,a | | EF |
| F0 | 240 | 'CURSOR up | [>=] | ret p | set 6,b | | F0 |
| F1 | 241 | 'CURSOR down | [<] | pop af | set 6,c | | F1 |
| F2 | 242 | 'CURSOR left | [<>] | jp p,**** | set 6,d | | F2 |
| F3 | 243 | 'CURSOR right | [<=] | di | set 6,e | | F3 |
| F4 | 244 | 'COPYCU up | [+] | call p,**** | set 6,h | | F4 |
| F5 | 245 | 'COPYCU down | [-] | push af | set 6,l | | F5 |
| F6 | 246 | 'COPYCU left | [*] | or ** | set 6,(hl) | | F6 |
| F7 | 247 | 'COPYCU right | [/] | rst 6 | set 6,a | | F7 |
| F8 | 248 | 'CURSOR SoTXT | [^] | ret m | set 7,b | | F8 |
| F9 | 249 | 'CURSOR EoTXT | [\] | ld sp,hl | set 7,c | | F9 |
| FA | 250 | 'CURSOR SoLN | [AND] | jp m,**** | set 7,d | | FA |
| FB | 251 | 'CURSOR EoLN | [MOD] | ei | set 7,e | | FB |
| FC | 252 | 'BREAK KEY | [OR] | call m,**** | set 7,h | | FC |
| FD | 253 | 'CAPS LOCK | [XOR] | | set 7,l | | FD |
| FE | 254 | 'SHIFT LOCK | [NOT] | cp ** | set 7,(hl) | | FE |
| FF | 255 | 'IGNORE | [TOKEN SWITCH] | rst 7 | set 7,a | | FF |

# Kleines Fachwörterbuch

**abandon** völlig aufgeben, verlassen
**adjust** anpassen, angleichen
**allocate** zuteilen, den Platz bestimmen
**arm** aufrüsten, wirksam machen
**background** Speicherbereich für externe Geräte
**base** Grundlage, Basis, Ausgangsadresse
**bottom** Grund, Untergrenze
**bound, boundary** Grenze, Begrenzung
**brandname** Firmenname
**chain** Kette, durch Verweise verbundene Blöcke
**column** Spalte, horizontale Cursorposition
**control code** Steuerzeichen
**count** Zähler
**CRTC, Cathode Ray Tube Controller** Bildschirmsteuerung
**current** hier: der laufende, der momentane
**default** Ersatzwert für fehlenden Parameter
**descriptor** Beschreiber, z.B. String Länge, Adresse
**destination** Ziel, Bestimmung
**device** Gerät, hier oft gleichbedeutend mit channel
**disable** unfähig machen, außerstand setzen
**disarm** entwaffnen, unwirksam machen
**dormant** schlafend, nicht bereite Task
**dot** Einzelpunkt (einer Matrix) auf dem Bildschirm
**enable** in den Stand setzen, befähigen
**entry** Einsprungadresse, Eintrag in einer Tabelle
**envelope** Hüllkurve, Umhüllung, Briefumschlag
**EoLN, end of line** gehe zum Ende der Zeile
**EoTXT, end of text** gehe zum Textende
**establish** einrichten, gründen
**evaluate** auswerten (Formel, Ausdruck)
**event** Ereignis, hier vergleichbar mit task
**expansion** Erweiterung, Ausdehnung (eines Zeichens)
**external** außerhalb, meist Zusatzgerät
**fail** schief gehen, erfolglos bleiben
**flag** Marke, Zeichen, kennzeichnet einen Zustand
**flashing period** Blink-Periode (einer Farbe)
**flood** fluten, Bereich mit einer Farbe auffüllen

**force** zwingen (Cursor ins Fenster)
**foreground** Speicherbereich für Anwenderprogramm
**garbage collect** 'Müllabfuhr', Strings neu ordnen
**indirection** Umleitung (ermöglicht Abfrage)
**initial value** Ausgangswert
**initialise** beginnen, einführen
**insert** einfügen (dabei den Rest verschieben)
**interrupt disable** Unterbrechungen verhindern
**interrupt** Unterbrechung des normalen Programmlaufs
**invalid** unvollständig, ungültig
**kernel** Kern, Grund-Funktionen des Betriebssystems
**legal** der Vorschrift entsprechend, zulässig
**limits** Grenzwerte (z. B. max. Zeilenanzahl)
**link** Verweis auf einen (nächsten) Block
**location** (Speicher-)Stelle
**log** eintragen (Logbuch), hier: Tabelle
**map** (Land-)Karte, Plan, meist Bit-Tabelle
**match** zueinander passen, übereinstimmen
**message** Nachricht, Mitteilung
**noisy** geräuschvoll, (bei CAS eher sichtbar)
**occur** sich ereignen, vorkommen
**offset** Versatz (zu einer Ausgangsadresse, base)
**overwrite** alten Text mit neuem überschreiben
**parameter** Übergabewert an ein Unterprogramm
**patch** Ändern (Ausbessern) einer Speicherstelle
**perform** durchführen
**pixel** Fachausdruck für 'dot', Einzelpunkt
**pointer** Zeiger (auf eine Speicherstelle)
**power** Potenz (mathematisch)
**prepare** vorbereiten auf
**present** gegenwärtig, augenblicklich
**preserve** bewahren, erhalten (meist Bits)
**previous** vorhergehend, bisheriger
**proceede** weitergehen, weitermachen
**prohibit** verbieten
**PSG** Programmable Sound Generator
**queue** Warteschlange
**remove** entfernen (des Cursors, aus einer Liste)

**reset** zurücksetzen (in Ausgangszustand, meist 0)
**row** Reihe, (Zeile), vertikale Cursorposition
**RSX, Resident System Extension** feste Erweiterung
**search** suchen (in einer Tabelle)
**select** auswählen, anwählen (ROM)
**sequence** Folge, Reihenfolge
**skip** überspringen, übergehen, auslassen
**soldered jumpers** Lötbrücken
**SoLN, start of line** gehe zum Zeilenanfang
**SoTXT, start of text** gehe zum Textanfang
**source** Quelle
**specify** genau aufzählen, anführen, angeben
**state map** Zustands-Tabelle (meist nur Bits)
**store** speichern, Speicher, (Memory)
**table** Tabelle, Liste
**task** Routine die gleichzeitig mit anderen abläuft
**temporary** vorübergehend, zeitweise
**terminated** beendet, abgeschlossen, fertig
**unchanged** unverändert
**undraw** wegnehmen, löschen (Cursor)
**unpack** eigentlich expand (char matrix)
**valid** gültig, zulässig
**validate** für gültig erklären, bestätigen
**VDU** Video Display Unit, Bildschirm
**vector** Zeiger (auf eine Speicherstelle, Liste)
**verify** auf Richtigkeit überprüfen

# Feed-Back Sheet (Rückkopplungszettel)

Das ist mein Urteil:
(bitte ankreuzen und/oder ausfüllen)

O Ich habe es bereits bedauert, das Geld für dieses Buch ausgegeben zu haben. Ich kann damit nichts anfangen.

O Warum müssen die Kommentare unbedingt in englisch sein? Ohne Wörterbuch komme ich da nicht klar.

O Ich finde dieses Buch nicht schlecht, es sollte aber mehr enthalten über: ...........................................
...........................................
...........................................
...........................................
...........................................

O Was mir gut gefällt: ...........................................
...........................................
...........................................
...........................................

O Was mir nicht so gut gefällt: ...........................................
...........................................
...........................................
...........................................

O Ich habe folgende Fehler entdeckt:
Seite/Adresse: ...........................................

Es muß richtig heißen: ...........................................
...........................................
...........................................
...........................................

Bitte an den Verlag S. Huslik, Postfach 10 18 24, 8900 Augsburg 1, einsenden. Ich freue mich, Ihr Urteil zu lesen.

Winfried Huslik

Weitere Mitteilungen oder Vorschläge:

Verlag S. Huslik
Postfach 10 18 24

8900 Augsburg 1

Absender:

## Feed-Back Sheet (Rückkopplungszettel)

Das ist mein Urteil:
(bitte ankreuzen und/oder ausfüllen)

O Ich habe es bereits bedauert, das Geld für dieses Buch ausgegeben zu haben. Ich kann damit nichts anfangen.

O Warum müssen die Kommentare unbedingt in englisch sein? Ohne Wörterbuch komme ich da nicht klar.

O Ich finde dieses Buch nicht schlecht, es sollte aber mehr enthalten über: ..............................
..............................
..............................
..............................
..............................

O Was mir gut gefällt: ..............................
..............................
..............................
..............................

O Was mir nicht so gut gefällt: ..............................
..............................
..............................
..............................

O Ich habe folgende Fehler entdeckt:
Seite/Adresse: ..............................

Es muß richtig heißen: ..............................
..............................
..............................
..............................

Bitte an den Verlag S. Huslik, Postfach 10 18 24, 8900 Augsburg 1, einsenden. Ich freue mich, Ihr Urteil zu lesen.

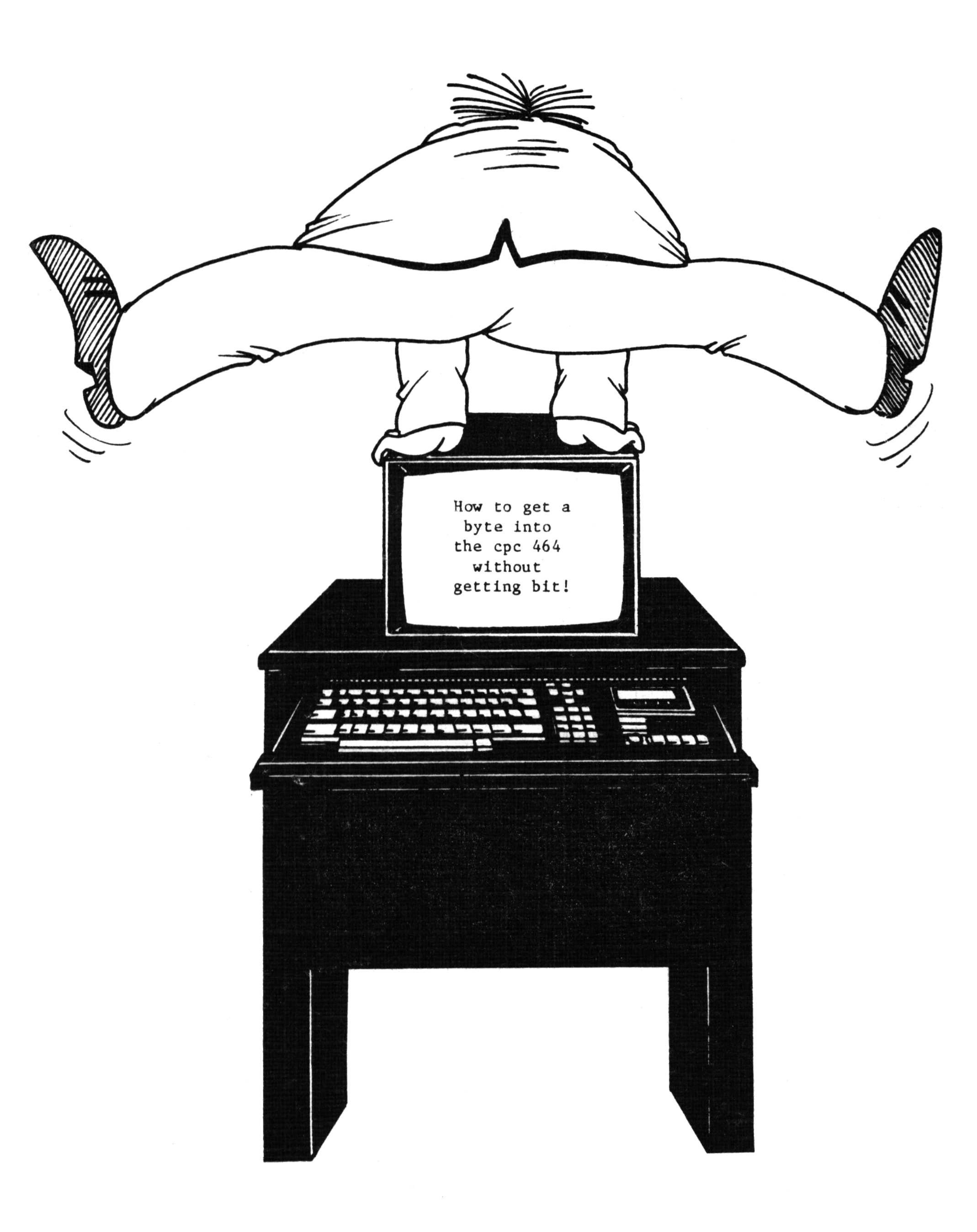
Winfried Huslik

Weitere Mitteilungen oder Vorschläge:

Verlag S. Huslik
Postfach 10 18 24

8900 Augsburg 1

Absender:




ISBN 3-925159-00-2 DM 59,00

Huslik

# CPC464 inside out

# AMSTRAD CPC

## MÉMOIRE ÉCRITE
## MEMORY ENGRAVED
## MEMORIA ESCRITA

https://acpc.me/